To develop this document is programming in the large.

To transcribe this document into compilable C computer programs is programming in the small.

To program in the small, the computer programs must not violate this document.

Tim Riley ©2000-2026

Appaserver Directory Tree

Configuration files:
- /etc/appaserver.config
- /etc/appaserver_$application.config

Source code directories:
- APPASERVER_HOME=/var/appaserver
- $APPASERVER_HOME/library
- $APPASERVER_HOME/utility
- $APPASERVER_HOME/javascript
- $APPASERVER_HOME/src_appaserver
- $APPASERVER_HOME/src_system
- $APPASERVER_HOME/src_$application

Data directories:
- /var/appaserver_data/$application
- /var/appaserver_data/$application/.texlive2021 -->
  - $APPASERVER_HOME/zfonts

Upload directories:
- /var/appaserver_upload/$application

Log files:
- /var/log/appaserver/appaserver_$application.err

Backup configuration files:
- /var/backups/appaserver/mysqldump_$application.config

Backup directories:
- /var/backups/appaserver/$application/son
- /var/backups/appaserver/$application/father
- /var/backups/appaserver/$application/grandfather
- /var/backups/appaserver/$application/greatgrandfather
- /var/backups/appaserver/$application/greatgreatgrandfather

Document root:
- DOCUMENT_ROOT=/var/www
- $DOCUMENT_ROOT/appaserver_home --> $APPASERVER_HOME
- $DOCUMENT_ROOT/appaserver_data --> /var/appaserver_data
- $DOCUMENT_ROOT/zmenu --> $APPASERVER_HOME/zmenu
- $DOCUMENT_ROOT/zscal2 --> $APPASERVER_HOME/zscal2
- $DOCUMENT_ROOT/zimages --> $APPASERVER_HOME/zimages
- $DOCUMENT_ROOT/appaserver/template/style.css -->
  - $APPASERVER_HOME/template/style.css
- $DOCUMENT_ROOT/appaserver/$application/index.php
- $DOCUMENT_ROOT/javascript --> $APPASERVER_HOME/javascript <-- to retire

Cloudacus.com:
- $CLOUDACUS_COM=$DOCUMENT_ROOT/cloudacus_com
- appaserver.config->document_root = $CLOUDACUS_COM
- $CLOUDACUS_COM/appaserver_home -> $APPASERVER_HOME
- $CLOUDACUS_COM/appaserver_data --> /var/appaserver_data
- $CLOUDACUS_COM/zmenu --> $APPASERVER_HOME/zmenu
- $CLOUDACUS_COM/zscal2 --> $APPASERVER_HOME/zscal2
- $CLOUDACUS_COM/zimages --> $APPASERVER_HOME/zimages
- $CLOUDACUS_COM/appaserver/template/style.css -->
  - $APPASERVER_HOME/template/style.css
- $CLOUDACUS_COM/appaserver/$application/index.php
- $CLOUDACUS_COM/images
- $CLOUDACUS_COM/.well-known <-- for TLS certificate verification
- $CLOUDACUS_COM/javascript --> $APPASERVER_HOME/javascript <-- to retire

APPASERVER_USER (1)

```
/* Usage */
APPASERVER_USER *appaserver_user_login_fetch(
    char *login_name,
    boolean fetch_role_name_list );

/* Process */
boolean entity_contact_key_boolean(
    ENTITY_TABLE,
    ENTITY_CONTACT_KEY_COLUMN );

char *entity_select_string(
    APPASERVER_USER_SELECT /* ENTITY_SELECT */,
    APPASERVER_USER_CONTACT_KEY_COLUMN,
    entity_contact_key_boolean() );

char *appaserver_user_login_where(
    APPASERVER_USER_LOGIN_NAME,
    login_name );

char *appaserver_system_string(
    entity_select_string(),
    APPASERVER_USER_TABLE,
    appaserver_user_login_where() );

char *string_system_input(
    appaserver_system_string() );

if ( !string_system_input() ) exit( 1 );

APPASERVER_USER *appaserver_user_parse(
    fetch_role_name_list,
    entity_contact_key_boolean(),
    string_system_input() );

/* Usage */
APPASERVER_USER *appaserver_user_parse(
    boolean fetch_role_name_list,
    boolean entity_contact_key_boolean,
    char *input );

/* Process */
APPASERVER_USER *appaserver_user_new(
    strdup( full_name ) );

if ( entity_contact_key_boolean )
{
    piece( buffer[], SQL_DELIMITER, input, 5 );
    appaserver_user_new()->contact_key = strdup( buffer );
}

if ( fetch_role_name_list )
{
    LIST *role_appaserver_user_name_list(
        ROLE_APPASERVER_USER_TABLE,
        appaserver_user_new()->full_name,
        appaserver_user_new()->contact_key,
        entity_contact_key_boolean() );
}

/* Usage */
char *appaserver_user_login_where(
    const char *APPASERVER_USER_LOGIN_NAME,
    char *login_name );

/* Process */
char *security_sql_injection_escape(
    SECURITY_ESCAPE_CHARACTER_STRING,
    login_name );

snprintf(
    where[],
    sizeof ( where ),
    "%s = '%s'",
    APPASERVER_USER_LOGIN_NAME,
    security_sql_injection_escape() );

/* Usage */
APPASERVER_USER *appaserver_user_new(
    char *full_name );

/* Process */
APPASERVER_USER *appaserver_user_calloc(
    void );

/* Usage */
APPASERVER_USER *appaserver_user_fetch(
    char *full_name /* stack memory */,
    char *contact_key /* stack memory */,
    boolean entity_contact_key_boolean,
    boolean fetch_role_name_list );

/* Process */
char *entity_select_string(
    APPASERVER_USER_SELECT /* ENTITY_SELECT */,
    APPASERVER_USER_CONTACT_KEY_COLUMN,
    entity_contact_key_boolean );

char *entity_primary_where(
    APPASERVER_FULL_NAME_COLUMN,
    APPASERVER_CONTACT_KEY_COLUMN,
    full_name,
    contact_key,
    entity_contact_key_boolean );

char *appaserver_system_string(
    entity_select_string(),
    APPASERVER_USER_TABLE,
    entity_primary_where() );

char *string_system_input(
    appaserver_system_string() );

if ( !string_system_input() ) return null;

APPASERVER_USER *appaserver_user_parse(
    fetch_role_name_list,
    entity_contact_key_boolean,
    string_system_input() );

/* Attributes */
char *full_name;
char *contact_key /* optional */;
char *login_name;
char *password;
char *user_date_format;
boolean deactivated_boolean;
LIST *role_appaserver_user_name_list;

/* Set externally */

/* Forbidden characters are \ and " */
char *typed_in_password;

char *encrypted_password;
enum password_function password_function;
boolean appaserver_user_password_encrypted;
```

APPASERVER_USER (2)

```
/* Usage */
char *appaserver_user_date_format_string(
    char *application_name,
    char *login_name );

/* Process */
char *user_date_format_string = “international”;

APPASERVER_USER *appaserver_user =
    appaserver_user_login_fetch(
        login_name,
        0 /* not fetch_role_name_list */ );

if ( appaserver_user->user_date_format )
{
    char *user_date_format_string =
        appaserver_user->user_date_format;
}
else
{
    APPLICATION *application =
        application_fetch(
            application_name );

    if ( application->user_date_format )
    {
        char *user_date_format_string =
            application->user_date_format;
    }
}
```

```
/* Usage */
char *appaserver_user_date_convert_string(
    char *appaserver_user_date_format_string(),
    char *international_date_string );

/* Process */
enum date_convert_format_enum
    date_convert_format_string_evaluate(
        appaserver_user_date_format_string );

DATE_CONVERT *date_convert =
    date_convert_new(
        date_convert_international
            /* source_enum */,
        date_convert_format_string_evaluate()
            /* destination_enum */,
        international_date_string
            /* source_date_string */ );

return date_convert->return_date_string;

/* Usage */
LIST *appaserver_user_role_name_list(
    char *login_name );

/* Process */
APPASERVER_USER *appaserver_user_login_fetch(
    login_name,
    1 /* fetch_role_name_list */ );
```

```
appaserver_user_login_fetch()->
    role_appaserver_user_name_list;

/* Usage */
void appaserver_user_update(
    const char *APPASERVER_USER_TABLE,
    char *password,
    char *login_name );

/* Process */
FILE *appaserver_user_update_open(
    const char *APPASERVER_USER_TABLE,
    const char *APPASERVER_USER_LOGIN_NAME );

char *security_mysql_version();

char *security_mysql_version_password_function(
    security_mysql_version() );

char *security_encrypt_password(
    password,
    security_mysql_version_password_function() );

void appaserver_user_update_pipe(
    FILE *appaserver_user_update_open(),
    char *security_encrypt_password(),
    char *login_name );

void pclose( appaserver_user_update_open() );
```

APPASERVER_ERROR (1)

```
/* Usage */
APPASERVER_ERROR *appaserver_error_new(
    char *application_name );

/* Process */
APPASERVER_ERROR *appaserver_error_calloc(
    void );

char *appaserver_error_filename(
    application_name );

/* Usage */
char *appaserver_error_filename(
    char *application_name );

/* Process */
if ( !application_name )
{
    application_name =
        environment_application_name()
}

APPASERVER_PARAMETER *
    appaserver_parameter =
        appaserver_parameter_application(
            application_name );

char *application_log_filename(
    APPLICATION_LOG_EXTENSION,
    application_name,
    appaserver_parameter->log_directory );

char *strdup( application_log_filename() );

/* Usage */
char *appaserver_error_prompt(
    char *argv_0,
    char *login_name );

/* Process */
static char prompt[];
char *ptr = prompt;

int date_utc_offset();
char *date_now_yyyy_mm_dd( date_utc_offset() );
char *date_now_hhmm( date_utc_offset() );

ptr += sprintf( ptr,
    "%s %s",
    date_now_yyyy_mm_dd(),
    date_now_hhmm() );

char *remote_ip_address =
    environment_get( "REMOTE_ADDR" );

if ( remote_ip_address )
{
    ptr += sprintf( ptr,
        " %s",
        remote_ip_address );
}

if ( login_name )
{
    ptr += sprintf( ptr,
        " %s",
        login_name );
}

ptr += sprintf( ptr, ":" );

if ( argv_0 )
{
    ptr += sprintf( ptr,
        " %s",
        argv_0 );
}

/* Usage */
FILE *appaserver_error_open_file(
    char *application_name );

/* Process */
char *appaserver_error_filename(
    application_name );

FILE *fopen( appaserver_error_filename(), "a" );

/* Attribute */
char *filename;
```

APPASERVER_ERROR (2)

```
/* Usage */
void appaserver_error_stderr(
    int argc,
    char **argv );

/* Process */
char *appaserver_error_prompt(
    *argv,
    (char *)0 /* login_name */ );

fprintf(
    stderr,
    "%s",
    appaserver_error_prompt() );

while ( --argc ) fprintf( stderr, " %s", *++argv );
fprintf( stderr, "\n" );
fflush( stderr );

/* Usage */
void appaserver_error_argv_file(
    int argc,
    char **argv,
    char *application_name,
    char *login_name );

/* Process */
FILE *file =
    appaserver_error_open_file(
        application_name );

char *appaserver_error_prompt(
    *argv,
    login_name );

fprintf(
    file,
    "%s",
    appaserver_error_prompt() );

while ( --argc ) fprintf( file, " %s", *++argv );
fprintf( file, "\n" );
fclose( file );

/* Usage */
void appaserver_error_message_file(
    char *application_name,
    char *login_name,
    char *message );

/* Process */
FILE *file =
    appaserver_error_open_file(
        application_name );

char *appaserver_error_prompt(
    (char *)0 /* argv_0 */,
    login_name );

fprintf(
    file,
    "%s %s\n",
    appaserver_error_prompt(),
    message );

fclose( file );

/* Usage */
void appaserver_error_stderr_exit(
    const char *file,
    const char *function,
    const int line,
    char *message );

/* Process */
sprintf(
    string[],
    "ERROR in %s/%s()/%d: %s",
    file,
    function,
    line,
    message );

fprintf( stderr, "%s\n", string );

char *environment_application_name();

if ( environment_application_name() )
{
    document_process_output(
        environment_application_name();
        (LIST *)0 /* javascript_filename_list */,
        "Error Occurred" /* title_string */ );

    printf(
        "<h3>%s</h3>\n",
        string );

    document_close();
}

exit( 1 );

/* Usage */
void msg(
    char *application_name,
    char *message );

/* Process */
appaserver_error_message_file(
    application_name,
    (char *)0 /* login_name */,
    message );

/* Usage */
void appaserver_error_argv_append_file(
    int arg,
    char **argv,
    char *application_name );

/* Process */
void appaserver_error_argv_file(
    argc,
    argv,
    application_name,
    (char *)0 /* login_name */ );

/* Usage */
void appaserver_error_login_name_append_file(
    int arg,
    char **argv,
    char *application_name,
    char *login_name );

/* Process */
void appaserver_error_argv_file(
    argc,
    argv,
    application_name,
    login_name );
```

APPASERVER

```
/* Usage */
char *appaserver_select(
    LIST *folder_attribute_name_list() );

/* Process */
char *list_delimited( folder_attribute_name_list, ‘,’ );

/* Usage */
char *appaserver_spreadsheet_heading_string(
    LIST *folder_attribute_name_list );

/* Process */
if ( !list_length( folder_attribute_name_list ) )
        return (char *)0;

list_double_quote_comma_display(
    char destination[],
    folder_attribute_name_list );

return strdup( destination );

/* Usage */
char *appaserver_spreadsheet_output_system_string(
    char delimiter,
    char *spreadsheet_filename );

/* Process */
sprintf(
    static char system_string[],
    “double_quote_comma_delimited.e ‘%c’ >> %s”,
    delimiter,
    spreadsheet_filename );

/* Usage */
char *appaserver_system_string(
    char *select,
    char *table,
    char *where_string );

/* Process */

/* Usage */
FILE *appaserver_input_pipe(
    char *appaserver_system_string() );

/* Process */

/* Usage */
FILE *appaserver_output_file(
    char *filename );

/* Process */

/* Usage */
FILE *appaserver_output_pipe(
    char *output_system_string );

/* Process */

/* Usage */
char *appaserver_update_filespecification(
    char *application_name,
    char *folder_name,
    char *session_key,
    pid_t process_id,
    LIST *relation_foreign_key_list,
    char *appaserver_parameter_data_directory );

/* Process */
char *appaserver_foreign_key_file_component(
    relation_foreign_key_list );

snprintf(
    update_filespecification[],
    sizeof ( update_filespecification ),
    “%s/%s/%s%s_%s_%d.dat”,
    appaserver_parameter_data_directory,
    application_name,
    folder_name,
    appaserver_foreign_key_file_component(),
    session_key,
    process_id );

/* Usage */
char *appaserver_mnemonic(
    LIST *key_list );

/* Process */
char *list_delimited( key_list, ‘_’ );

/* Usage */
char *appaserver_table_name(
    char *folder_name );

/* Process */
application_table_name( folder_name );

/* Usage */
char *appaserver_mailname(
    const char *
        APPASERVER_MAILNAME_FILESPECIFICATION );

/* Usage */
boolean appaserver_executable_exists_boolean(
    char *executable );

/* Process */

/* Usage */
char *appaserver_foreign_key_file_component(
    LIST *relation_foreign_key_list );

/* Process */
if ( list_length( relation_foreign_key_list ) )
{
    char *appaserver_mnemonic(
        relation_foreign_key_list );

    snprintf(
        static foreign_key_file_component[],
        sizeof ( file_component ),
        “_%s”,
        appaserver_mnemonic() );
}
else
{
    strcpy( static foreign_key_file_component[], “” );
}
```

ENTITY (1)

```
/* Usage */
ENTITY *entity_fetch(
    char *full_name,
    char *contact_key,
    boolean entity_contact_key_boolean );

/* Process */
static char *entity_primary_where(
    ENTITY_FULL_NAME_COLUMN,
    ENTITY_CONTACT_KEY_COLUMN,
    full_name,
    contact_key,
    entity_contact_key_boolean );

char *entity_select_string(
    ENTITY_SELECT,
    ENTITY_CONTACT_KEY_COLUMN,
    entity_contact_key_boolean() );

char *appaserver_system_string(
    entity_select_string(),
    const char *ENTITY_TABLE,
    char *entity_primary_where() );

char *string_system_input(
    appaserver_system_string() );

free( appaserver_system_string() );

ENTITY *entity_parse(
    entity_contact_key_boolean(),
    string_system_input() );

free( string_system_input() );

/* Usage */
ENTITY *entity_parse(
    boolean entity_contact_key_boolean,
    char *input );

/* Process */
if ( entity_contact_key_boolean )
{
    piece( buffer[], SQL_DELIMITER, input, 8 );
    char *contact_key = strdup( buffer );
}

ENTITY *entity_new(
    entity_contact_key_boolean,
    strdup( full_name ),
    contact_key );

/* Usage */
ENTITY *entity_new(
    boolean entity_contact_key_boolean,
    char *full_name,
    char *contact_key );

/* Process */
ENTITY *entity_calloc(
    void );

/* Usage */
char *entity_select_string(
    const char *ENTITY_SELECT,
    const char *ENTITY_CONTACT_KEY_COLUMN,
    boolean entity_contact_key_boolean );

/* Process */
OPTIONAL_COLUMN *optional_column_new(
    ',' /* delimiter */,
    ENTITY_SELECT /* base_string */,
    ENTITY_CONTACT_KEY_COLUMN /* component */,
    0 /* not escape_boolean */,
    entity_contact_key_boolean /* set_boolean */ );

heap char *optional_column_new()->return_string;

/* Attributes */
boolean entity_contact_key_boolean;
char *full_name;
char *contact_key /* optional */,
char *street_address;
char *city;
char *state_code;
char *zip_code;
char *land_phone_number;
char *cell_phone_number;
char *email_address;
```

ENTITY (2)

```
/* Usage */
static char *entity_primary_where(
    const char *ENTITY_FULL_NAME_COLUMN,
    const char *ENTITY_CONTACT_KEY_COLUMN,
    char *table_name,
    char *full_name,
    char *contact_key,
    boolean entity_contact_key_boolean );

/* Process */
boolean entity_full_name_populated_boolean(
    ENTITY_FULL_NAME_COLUMN,
    full_name );

if ( !entity_full_name_poulated_boolean() )
{
    strcpy( static where[], "1 = 1" );
}
else
{
    static char *entity_table_name_column(
        ENTITY_FULL_NAME_COLUMN
            /* column_name */,
        table_name );

    char *entity_escape_full_name(
        full_name );

    static char *entity_contact_key_where(
        ENTITY_CONTACT_KEY_COLUMN,
        table_name,
        contact_key,
        entity_contact_key_boolean );

    snprintf(
        static where[],
        sizeof ( where ),
        "%s = '%s' and "
        "%s",
        entity_table_name_column(),
        entity_escape_full_name(),
        entity_contact_key_where() );

    free( entity_escape_full_name() );
}

/* Usage */
char *entity_join(
    const char *ENTITY_TABLE,
    const char *FOREIGN_TABLE,
    const char *ENTITY_FULL_NAME_COLUMN,
    const char *ENTITY_CONTACT_KEY_COLUMN,
    boolean entity_contact_key_boolean );

/* Process */
char join[];
char *ptr = join;

strcpy(
    entity_column,
    static char *entity_table_name_column(
        ENTITY_FULL_NAME_COLUMN,
        ENTITY_TABLE ) );

strcpy(
    foreign_column,
    static char *entity_table_name_column(
        ENTITY_FULL_NAME_COLUMN,
        FOREIGN_TABLE ) );

ptr += sprintf( ptr,
    "%s = %s",
    entity_column,
    foreign_column );

if ( entity_contact_key_boolean )
{
    strcpy(
        entity_column,
        static char *entity_table_name_column(
            ENTITY_CONTACT_KEY_COLUMN,
            ENTITY_TABLE ) );

    strcpy(
        foreign_column,
        static char *entity_table_name_column(
            ENTITY_CONTACT_KEY_COLUMN,
            FOREIGN_TABLE ) );

    ptr += sprintf( ptr,
        " and %s = %s",
        entity_column,
        foreign_column );
}

/* Usage */
char *entity_escape_full_name(
    char *full_name );

/* Process */
if ( !full_name || !*full_name ) exit( 1 );

char *security_sql_injection_escape(
    SECURITY_ESCAPE_CHARACTER_STRING,
    full_name );

/* Usage */
char *entity_escape_contact_key(
    char *contact_key );

/* Process */
if ( !contact_key || !*contact_key ) exit( 1 );

char *security_sql_injection_escape(
    SECURITY_ESCAPE_CHARACTER_STRING,
    contact_key );
```

ENTITY (3)

```
/* Usage */
static char *entity_contact_key_where(
    const char *ENTITY_CONTACT_KEY_COLUMN,
    char *table_name,
    char *contact_key,
    boolean entity_contact_key_boolean );

/* Process */
if (    !contact_key
||      !entity_contact_key_boolean )
{
    strcpy( static where[], "1 = 1" );
}
else
{
    static char *entity_table_name_column(
        ENTITY_CONTACT_KEY_COLUMN
            /* column_name ),
        table_name );

    char *entity_escape_contact_key(
        contact_key );

    snprintf(
        static where,
        sizeof ( where ),
        "%s = '%s'",
        entity_table_name_column(),
        entity_escape_contact_key() );

    free( entity_escape_contact_key() );
}

/* Usage */
static char *entity_table_name_column(
    const char *column_name,
    char *table_name );

/* Process */
if ( !table_name )
{
    strcpy( static table_name_column[], column_name );
}
else
{
    snprintf(
        static table_name_column[],
        sizeof ( table_name_column ),
        "%s.%s",
        table_name,
        column_name );
}

/* Usage */
char *entity_fetch_contact_key(
    boolean entity_contact_key_boolean,
    char *full_name );

/* Process */
if ( entity_contact_key_boolean )
{
    static char *entity_primary_where(
        ENTITY_FULL_NAME_COLUMN,
        ENTITY_CONTACT_KEY_COLUMN,
        (char *)0 /* table_name */,
        full_name,
        (char *)0 /* contact_key */ );
        1 /* entity_contact_key_boolean */ );

    char *appaserver_system_string(
        ENTITY_CONTACT_KEY_COLUMN /* select */,
        ENTITY_TABLE,
        entity_primary_where() );

    char *string_system_input(
        appaserver_system_string() );

    free( appaserver_system_string() );
}

/* Usage */
LIST *entity_full_contact_list(
    LIST *full_name_list,
    LIST *contact_key_list );

/* Process */
boolean entity_contact_key_boolean(
    ENTITY_TABLE,
    ENTITY_CONTACT_KEY_COLUMN );

ENTITY *entity_new(
    entity_contact_key_boolean,
    list_get( full_name_list ),
    list_get( contact_key_list ) );

/* Usage */
ENTITY *entity_full_name_seek(
    char *full_name,
    LIST *entity_list );

/* Process */

/* Usage */
ENTITY *entity_getset(
    boolean entity_contact_key_boolean,
    LIST *entity_list /* in/out */,
    char *full_name,
    char *contact_key,
    boolean strdup_boolean );

/* Process */

/* Usage */
ENTITY *entity_seek(
    char *full_name,
    char *contact_key,
    LIST *entity_list );

/* Process */

/* Usage */
boolean entity_list_exist_boolean(
    char *full_name,
    char *contact_key,
    LIST *entity_list );

/* Process */
```

ENTITY (4)

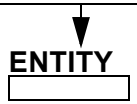
```
/* Usage */
char *entity_insert(
    const char *ENTITY_TABLE,
    const char *ENTITY_FULL_NAME_COLUMN,
    const char *ENTITY_CONTACT_KEY_COLUMN,
    boolean entity_contact_key_boolean,
    char *full_name,
    char *contact_key,
    boolean ignore_duplicate_boolean );

/* Process */
char *entity_insert_system_string(
    ENTITY_TABLE,
    ENTITY_FULL_NAME_COLUMN,
    ENTITY_CONTACT_KEY_COLUMN,
    SQL_DELIMITER,
    entity_contact_key_boolean,
    ignore_duplicate_boolean );

char *entity_primary_data_string(
    SQL_DELIMITER,
    entity_contact_key_boolean,
    full_name,
    contact_key );

char *spool_data_string(
    entity_insert_system_string(),
    entity_primary_data_string() );

free( entity_insert_system_string() );
free( entity_primary_data_string() );

/* Usage */
char *entity_insert_system_string(
    const char *ENTITY_TABLE,
    const char *ENTITY_FULL_NAME_COLUMN,
    const char *ENTITY_CONTACT_KEY_COLUMN,
    const char SQL_DELIMITER,
    boolean entity_contact_key_boolean,
    boolean ignore_duplicate_boolean );

/* Process */
char *entity_insert_column_string(
    ENTITY_FULL_NAME_COLUMN /* ENTITY_INSERT */,
    ENTITY_CONTACT_KEY_COLUMN,
    entity_contact_key_boolean );

snprintf(
    system_string[],
    sizeof ( system_string ),
    "insert_statement table=%s field=%s delimiter='%c' | "
    "sql 2>&1 | "
    "%s",
    ENTITY_TABLE,
    entity_insert_column_string(),
    SQL_DELIMITER,
    (ignore_duplicate_boolean)
        ? "grep -vi duplicate"
        : "cat" );

free( entity_insert_column_string() );
char *strdup( system_string );

/* Usage */
char *entity_insert_column_string(
    const char *ENTITY_INSERT,
    const char *ENTITY_CONTACT_KEY_COLUMN,
    boolean entity_contact_key_boolean );

/* Process */
OPTIONAL_COLUMN *optional_column_new(
    ',' /* delimiter */,
    ENTITY_FULL_NAME_COLUMN /* base_string */,
    ENTITY_CONTACT_KEY_COLUMN /* component */,
    0 /* not escape_boolean */,
    entity_contact_key_boolean /* set_boolean */ );

heap char *optional_column_new()->return_string;

/* Usage */
char *entity_primary_data_string(
    const char delimiter,
    boolean entity_contact_key_boolean,
    char *full_name,
    char *contact_key );

/* Process */
OPTIONAL_COLUMN *optional_column_new(
    delimiter,
    full_name /* base_string */,
    contact_key /* component */,
    1 /* escape_boolean */,
    entity_contact_key_boolean /* set_boolean */ );

optional_column_new()->return_string /* heap memory */;

/* Usage */
static char *entity_name_display(
    const char delimiter,
    char *full_name,
    char *contact_key,
    boolean entity_contact_key_boolean );

/* Process */

/* Usage */
ENTITY *entity_full_name_entity(
    boolean entity_contact_key_boolean,
    char *full_name /* stack memory */ );

/* Process */
if ( !full_name ) return null;

char *entity_fetch_contact_key(
    entity_contact_key_boolean,
    full_name );

ENTITY *entity_new(
    entity_contact_key_boolean,
    strdup( full_name ),
    entity_fetch_contact_key() );

/* Usage */
boolean entity_contact_key_boolean(
    const char *ENTITY_TABLE,
    const char *ENITTY_CONTACT_KEY_COLUMN );

/* Process */
static boolean contact_key_boolean = -1;

if ( contact_key_boolean == -1 )
{
    boolean appaserver_table_column_boolean(
        ENTITY_TABLE /* table_name */,
        ENTITY_CONTACT_KEY_COLUMN /* column_name */);
}
```

ENTITY (5)

```
/* Usage */
char *entity_contact_key(
    const char *ENTITY_TABLE,
    const char *ENTITY_CONTACT_KEY_COLUMN,
    const char *ENTITY_CONTACT_KEY_DEFAULT,
    char *full_name,
    char *contact_key,
    boolean entity_contact_key_boolean );

/* Process */
char *key = {0};

if ( entity_contact_key_boolean )
{
    boolean entity_contact_key_populated_boolean(
        ENTITY_CONTACT_KEY_COLUMN,
        contact_key );

    if ( entity_contact_key_populated_boolean()
    {
        char *key = contact_key;
    }
    else
    {
        char *key =
            entity_fetch_contact_key(
                1 /* entity_contact_key_boolean */,
                full_name );

        if ( !key )
        {
            char *key =
                folder_attribute_fetch_default_value(
                    ENTITY_TABLE
                        /* folder_name */,
                    ENTITY_CONTACT_KEY_COLUMN
                        /* attribute_name */ );
        }

        if ( !key )
        {
            key = ENTITY_CONTACT_KEY_DEFAULT;
        }
    }
}

/* Usage */
boolean entity_contact_key_populated_boolean(
    const char *ENTITY_CONTACT_KEY_COLUMN,
    char *contact_key );

/* Process */
if (    contact_key
&&  *contact_key
&&  strcmp(
        contact_key,
        ENTITY_CONTACT_KEY_COLUMN ) != 0 )
{
    populated_boolean = 1;
}

/* Usage */
boolean entity_full_name_populated_boolean(
    const char *ENTITY_FULL_NAME_COLUMN,
    char *full_name );

/* Process */
boolean entity_contact_key_populated_boolean(
    ENTITY_FULL_NAME_COLUMN
        /* ENTITY_CONTACT_KEY_COLUMN */,
    full_name
        /* contact_key */ );

/* Usage */
LIST *entity_primary_key_list(
     const char *ENTITY_FULL_NAME_COLUMN,
     const char *ENTITY_CONTACT_KEY_COLUMN,
     boolean entity_comtact_key_boolean );

/* Process */
list_set( list, ENTITY_FULL_NAME_COLUMN );

if ( entity_contact_key_boolean )
{
     list_set( list, ENTITY_CONTACT_KEY_COLUMN );
}
```

ENTITY_SELF

```
/* Usage */
static ENTITY_SELF *entity_self_fetch(
    boolean entity_contact_key_boolean,
    boolean fetch_entity_boolean );

/* Process */
static ENTITY_SELF *entity_self;

if ( entity_self ) return entity_self;

char *entity_self_select(
    const char *ENTITY_SELF_SELECT,
    const char *ENTITY_CONTACT_KEY_COLUMN,
    boolean entity_contact_key_boolean );

char *appaserver_system_string(
    entity_self_select(),
    ENTITY_SELF_TABLE,
    (char *)0 /* where */ );

free( entity_self_select() );

char *string_system_input(
    appaserver_system_string() );

free( appaserver_system_string() );

ENTITY_SELF *entity_self =
    entity_self_parse(
        entity_contact_key_boolean,
        fetch_entity_boolean,
        string_system_input() );

free( string_system_input() );

if ( !entity_self_parse() ) exit( 1 );

/* Usage */
ENTITY_SELF *entity_self_parse(
    boolean entity_contact_key_boolean,
    boolean fetch_entity_boolean,
    char *input );

/* Process */
if ( entity_contact_key_boolean )
{
    piece( buffer[], SQL_DELIMITER, input, 7 );
    char *contact_key = strdup( buffer );
}

ENTITY_SELF *entity_self_new(
    entity_contact_key_boolean,
    strdup( full_name ),
    contact_key );

if ( fetch_entity_boolean )
{
    entity_self_new()->entity =
        entity_fetch(
            entity_self_new()->full_name,
            entity_self_new()->contact_key,
            entity_contact_key_boolean );

    if ( !entity_self_new()->entity ) exit( 1 );
}

/* Usage */
ENTITY_SELF *entity_self_new(
    boolean entity_contact_key_boolean,
    char *full_name,
    char *contact_key );

/* Process */
ENTITY_SELF *entity_self_calloc(
    void );

/* Attributes */
boolean entity_contact_key_boolean;
char *full_name;
char *contact_key /* optional */;
char *credit_card_number;
char *credit_card_expiration_month_year;
char *credit_card_security_code;
char *credit_provider;
double invoice_amount_due;
char *invoice_statement_current;
ENTITY *entity;
```



ENTITY_KEY

```
/* Usage */
ENTITY_KEY *entity_key_new(
    void );

/* Process */
ENTITY_KEY *entity_key_calloc(
    void );

/* Attributes */
```

SECURITY (1)

```
enum password_function password_function;

#define SECURITY_ESCAPE_CHARACTER_STRING
"~!`'$;&=()\""

/* Usage */
char *security_sql_injection_escape(
    const char *SECURITY_ESCAPE_CHARACTER_STRING,
    char *datum );

/* Process */
if ( !datum || !*datum ) return strdup( "" );

boolean string_character_boolean( datum, '\\' );

if ( string_character_boolean() )
        return strdup( datum );

char *strdup(
    char *string_escape_character_array(
            char destination[],
            datum,
            SECURITY_ESCAPE_CHARACTER_STRING ) );

/* Usage */
char *security_sql_injection_unescape(
    const char *SECURITY_ESCAPE_CHARACTER_STRING,
    char *datum );

/* Process */
char *strdup(
    char *string_unescape_character_array(
            char destination[],
            datum,
            SECURITY_ESCAPE_CHARACTER_STRING ) );

/* Usage */
char *security_replace_special_characters(
    char *datum );

/* Process */

/* Usage */
char *security_sql_injection_escape_quote_delimit(
    char *datum );

/* Process */

/* Usage */
char *security_generate_password(
    void );

/* Process */
string_pipe( "generate_password.sh" );

/* Usage */
unsigned long security_random(
    unsigned long up_to );

/* Process */
return
(unsigned long)( ( drand48() * up_to ) + 1 );

/* Usage */
char *security_remove_semicolon(
    char *datum );

/* Process */
int inside_quote = 0;
char destination[];
char *ptr = destination;

boolean string_character_boolean(
    datum, ';' );

if ( !string_character_boolean() ) return datum;

while ( *datum )
{
    if ( *datum == '\'' )
    {
        inside_quote = 1 - inside_quote;
    }

    if ( *datum == ';' )
    {
        if ( inside_quote )
        {
            *ptr++ = ';';
        }
    }
    else
    {
        *ptr++ = *datum;
    }
    datum++;
}

/* Usage */
boolean security_forbid_boolean(
    const char *SECURITY_FORBID_CHARACTER_STRING,
    char *string );

/* Process */
```

enum password_function

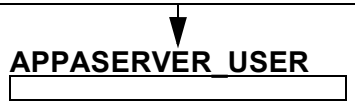
no_encryption
old_password_function
regular_password_function
sha2_function

SECURITY (2)

```
/* Usage */
char *security_encrypt_password(
    char *post_login_password,
    enum password_function password_function );

/* Process */
char *security_encrypt_select_clause(
    password_function,
    post_login_password );

snprintf(
    system_string[],
    sizeof ( system_string ),
    "select.sh \"%s\"",
    security_encryption_select_clause() );

char *spool_pipe(
    system_string,
    0 /* not capture_stderr_boolean */ );

/* Usage */
boolean security_password_match(
    char *appaserver_user_password,
    char *security_encrypt_password );

/* Process */
if (    !appaserver_user_password
||      !*appaserver_user_password )
        return 0;

if ( string_strcmp(
        appaserver_user_password,
        "null" ) == 0 )
        return 0;

( strcmp(
    security_encrypt_password,
    appaserver_user_password ) == 0 );

/* Usage */
char *security_encrypt_select_clause(
    enum password_function password_function,
    char *post_login_password );

/* Process */
char *string_escape_single_quotes(
    post_login_password );

if ( password_function == no_encryption )
{
    snprintf(
        select_clause[],
        sizeof ( select_clause ),
        "'%s'",
        string_escape_single_quotes() );
}
else
if ( password_function == old_password_function )
{
    snprintf(
        select_clause[],
        sizeof ( select_clause ),
        "old_password('%s')",
        string_escape_single_quotes() );
}
else
if ( password_function == regular_password_function )
{
    snprintf(
        select_clause[],
        sizeof ( select_clause ),
        "CONCAT('*',UPPER(SHP1(UNHEX(SHA1('%S'))))",
        string_escape_single_quotes() );
}
else
if ( password_function == sha2_function )
{
    snprintf(
        select_clause[],
        sizeof ( select_clause ),
        "sha2('%s',224)",
        string_escape_single_quotes() );
}
else
{
    exit( 1 );
}

/* Usage */
void security_system(
    const char SECURITY_FORK_CHARACTER,
    const char *SECURITY_FORK_STRING,
    char *system_string );

/* Process */
boolean string_character_boolean(
    system_string,
    SECURITY_FORK_CHARACTER );

boolean string_exists(
    system_string,
    SECURITY_FORK_STRING /* substring */ );

if (    string_character_boolean()
||      string_exists() )
{
    exit( 1 );
}

system( system_string );

/* Usage */
enum password_function
    security_password_function(
        char *appaserver_user_password );

/* Process */

/* Public */
char *security_mysql_version( void );

enum password_function
    security_mysql_version_password_function(
        security_mysql_version() );

boolean security_password_encrypted(
    char *password );

LIST *security_sql_injection_escape_list(
    LIST *data_list );
```

SECURITY_ENTITY

```
/* Usage */
SECURITY_ENTITY *security_entity_new(
    char *login_name,
    char *folder_name,
    boolean folder->
        folder_row_level_restriction->
        non_owner_forbid,
    boolean role->override_row_restrictions );

/* Process */
SECURITY_ENTITY *security_entity_calloc(
    void );

if (    !non_owner_forbid
||      override_row_restrictions )
{
    return this;
}

APPASERVER_USER *appaserver_user =
    appaserver_user_login_fetch(
        login_name,
        0 /* not fetch_role_name_list */ );

static/null char *security_entity_table_name(
    folder_name );

char *security_entity_where(
    security_entity_table_name(),
    appaserver_user->full_name,
    appaserver_user->contact_key );

/* Usage */
char *security_entity_where(
    char *table_name,
    char *full_name,
    char *contact_key );

/* Process */
boolean entity_contact_key_boolean(
    ENTITY_TABLE,
    ENTITY_CONTACT_KEY_COLUMN );

static char *entity_primary_where(
    APPASERVER_FULL_NAME_COLUMN,
    APPASERVER_CONTACT_KEY_COLUMN,
    table_name,
    full_name,
    contact_key,
    entity_contact_key_boolean() );

char *strdup( entity_primary_where() );

/* Usage */
static/null char *security_entity_table_name(
    char *folder_name );

/* Process */
(folder_name)
    ? static appaserver_table_name( folder_name )
    : (char *)0;

/* Attributes */
char *login_name;
boolean non_owner_forbid;
boolean override_row_restrictions;
APPASERVER_USER *appaserver_user;
char *where;
```



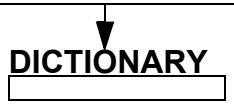
APPASERVER_USER

APPASERVER_PARAMETER

```
/* Usage */
APPASERVER_PARAMETER *
   appaserver_parameter_new(
      void );

/* Process */
char *environment_application_name();

if ( !environment_application_name() ) exit( 1 );

APPASERVER_PARAMETER *
   appaserver_parameter_application(
      environment_application_name() );

/* Usage */
APPASERVER_PARAMETER *
   appaserver_parameter_application(
      char *application_name );

/* Process */
APPASERVER_PARAMETER *
   appaserver_parameter_calloc(
      void );

char *appaserver_parameter_filename(
   char *application_name );

FILE *appaserver_parameter_open_file(
   char *appaserver_parameter_filename() );

DICTIONARY *appaserver_parameter_dictionary(
   appaserver_parameter_open_file() );

fclose( appaserver_parameter_open_file() );

APPASERVER_PARAMETER *
   appaserver_parameter_fetch(
      appaserver_parameter_application(),
      appaserver_parameter_dictionary() );

/* Usage */
DICTIONARY *appaserver_parameter_dictionary(
   FILE *file );

/* Process */
DICTIONARY *dictionary_new();

while ( string_input( input[], file ) )
{
   if ( *input == '[' ) continue;

   piece( key[], '=', input, 0 );
   piece( datum[], '=', input, 1 );

   if ( piece() )
   {
      dictionary_set(
         dictionary_new(),
         strdup( key ),
         strdup( datum ) );
   }
}

/* Attributes */
char *filename;
char *mysql_user;
char *password;
char *flags;
char *MYSQL_HOST;
char *MYSQL_TCP_PORT;
char *mount_point;
char *apache_cgi_directory;
char *cgi_home;
char *document_root;
char *log_directory;
char *data_directory;
char *upload_directory;
char *backup_directory;
boolean mysql_password_syntax_boolean;
boolean bypass_ip_changed_check_boolean;

/* Public */
char *appaserver_parameter_mount_point(
   void );

char *appaserver_parameter_cgi_directory(
   void );

char *appaserver_parameter_log_directory(
   void );

char *appaserver_parameter_data_directory(
   void );

char *appaserver_parameter_cgi_home(
   void );

char *appaserver_parameter_document_root(
   void );

char *appaserver_parameter_upload_directory(
   void );

char *appaserver_parameter_backup_directory(
   void );

boolean
   appaserver_parameter_bypass_ip_changed_check(
      void );
```

POST_DICTIONARY (1)

```
/* Usage */
POST_DICTIONARY *post_dictionary_stdin_new(
    /* Used when expecting a spooled file */
    char *WIDGET_UPLOAD_RECALL_PREFIX,
    char *application_name,
    char *appaserver_parameter->upload_directory,
    LIST *upload_filename_list );

/* Process */
POST_DICTIONARY *post_dictionary_calloc(
    void );

char *string_input( stdin );

char *post_dictionary_apache_key(
    string_input() );

if ( !post_dictionary_apache_key() )
{
    free( post_dictionary_calloc() );
    return
    POST_DICTIONARY *
        post_dictionary_string_new(
            DICTIONARY_ATTRIBUTE_DATUM_DELIMITER,
            DICTIONARY_ELEMENT_DELIMITER,
            string_input() /* dictionary_string */ );
}

DICTIONARY *original_post_dictionary =
    dictionary_huge();

while ( 1 )
{
    POST_DICTIONARY_ATTRIBUTE *
        post_dictionary_attribute_new(
            WIDGET_SELECT_OPERATOR,
            WIDGET_UPLOAD_RECALL_PREFIX,
            application_name,
            upload_directory,
            upload_filename_list,
            post_dictionary_apache_key() );

    if ( !post_dictionary_attribute_new() ) break;

    if (    post_dictionary_attribute_new()->name
    &&  post_dictionary_attribute_new()->datum )
    {
        void post_dictionary_datum_set(
            WIDGET_MULTI_LEFT_RIGHT_DELIMITER,
            original_post_dictionary /* out */,
            post_dictionary_attribute_new()->name,
            post_dictionary_attribute_new()->datum /* in/out */ );
    }
}

/* Attributes */
char *apache_key;
DICTIONARY *original_post_dictionary;
```

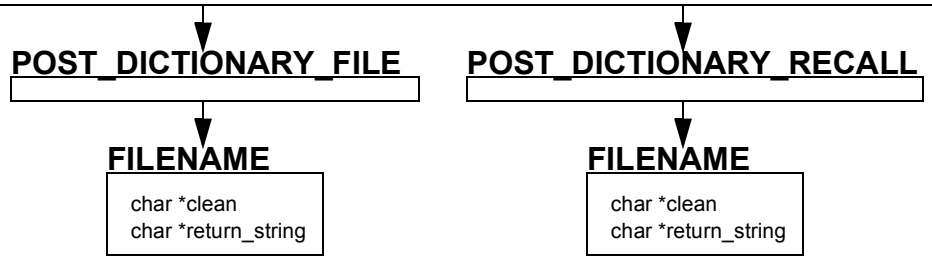


POST_DICTIONARY (2)

```
/* Usage */
char *post_dictionary_apache_key(
    char *string_input );

/* Process */
if ( string_input && *string_input == '-' )
    char *apache_key = strdup( string_input );

/* Usage */
void post_dictionary_datum_set(
    const char WIDGET_MULTI_LEFT_RIGHT_DELIMITER,
    DICTIONARY *original_post_dictionary /* out */,
    char *post_dictionary_attribute_name,
    char *post_dictionary_attribute_datum /* in/out */ );

/* Process */
/* Example: BA|1 returns 1 and datum = BA */
int post_dictionary_row_number(
    post_dictionary_attribute_datum /* in/out */,
    WIDGET_MULTI_LEFT_RIGHT_DELIMITER );

if ( post_dictionary_row_number() )
{
    char *post_dictionary_row_attribute_name(
        post_dictionary_attribute_name,
        post_dictionary_row_number() );

    free( post_dictionary_attribute_name );

    post_dictionary_attribute_name =
        post_dictionary_row_attribute_name();
}

dictionary_set(
    original_post_dictionary,
    post_dictionary_attribute_name,
    post_dictionary_attribute_datum );

/* Usage */
/* Example: BA|1 returns 1 and datum = BA */
int post_dictionary_row_number(
    char *datum /* in/out */,
    const char WIDGET_MULTI_LEFT_RIGHT_DELIMITER );

/* Process */
char *ptr = datum + strlen( datum ) - 1;

while ( ptr > datum )
{
    if ( isdigit( *ptr ) )
    {
        ptr--;
        continue;
    }

    if ( *ptr == WIDGET_MULTI_LEFT_RIGHT_DELIMITER )
    {
        int row_number = atoi( ptr + 1 ) + 1;
        *ptr = '\0';
        return row_number;
    }

    break;
}

return 0;

/* Usage */
POST_DICTIONARY *post_dictionary_string_new(
    const char *
        DICTIONARY_ATTRIBUTE_DATUM_DELIMITER,
    const char *
        DICTIONARY_ELEMENT_DELIMITER,
    char *dictionary_string );

/* Process */
POST_DICTIONARY *post_dictionary =
    post_dictionary_calloc();

post_dictionary->original_post_dictionary =
    dictionary_string_new(
        DICTIONARY_ATTRIBUTE_DATUM_DELIMITER,
        DICTIONARY_ELEMENT_DELIMITER,
        dictionary_string );

/* Usage */
char *post_dictionary_row_attribute_name(
    char *post_dictionary_attribute_name,
    int post_dictionary_row_number );

/* Process */
snprintf(
    row_attribute_name[],
    sizeof ( row_attribute_name ),
    "%s_%d",
    post_dictionary_attribute_name,
    post_dictionary_row_number );
```

POST_DICTIONARY_ATTRIBUTE (1)

```
/* Usage */
POST_DICTIONARY_ATTRIBUTE *
    post_dictionary_attribute_new(
        const char *WIDGET_SELECT_OPERATOR,
        char *WIDGET_UPLOAD_RECALL_PREFIX,
        char *application_name,
        char *upload_directory,
        LIST *upload_filename_list,
        char *post_dictionary_apache_key );

/* Process */
while ( string_input( input[], stdin, sizeof ( input ) ) )
{
    if ( !*input ) continue;

    static char *post_dictionary_attribute_apache_marker(
        input );

    if ( !post_dictionary_attribute_apache_marker() ) continue;

    /* New attribute */
    if ( strcmp(
        post_dictionary_apache_marker(),
        "Content-Disposition: form-data; name=" ) == 0 )
    {
        POST_DICTIONARY_ATTRIBUTE *
            post_dictionary_attribute_calloc(
                void );

        char *post_dictionary_attribute_name(
            input );

        POST_DICTIONARY_FILE *
            post_dictionary_file_new(
                POST_DICTIONARY_FILE_NAME_MAXSIZE,
                application_name,
                upload_directory,
                upload_filename_list,
                post_dictionary_apache_key,
                input,
                post_dictionary_attribute_name() );

        if ( post_dictionary_file_new() )
        {
            if ( post_dictionary_file_new()->filename )
            {
                this->datum =
                    post_dictionary_file_new()->
                        filename->
                        return_string;
            }
            return this;
        }

        POST_DICTIONARY_RECALL *
            post_dictionary_recall_new(
                WIDGET_UPLOAD_RECALL_PREFIX,
                POST_DICTIONARY_FILE_NAME_MAXSIZE,
                application_name,
                upload_directory,
                post_dictionary_apache_key,
                post_dictionary_attribute_name() );

        if ( post_dictionary_recall_new() )
        {
            if ( post_dictionary_recall_new()->datum )
            {
                this->name =
                    post_dictionary_recall_new()->
                    attribute_name;

                this->datum =
                    post_dictionary_recall_new()->
                        filename->
                        return_string;
            }
            return this;
        }

        char *post_dictionary_attribute_datum(
            WIDGET_SELECT_OPERATOR,
            stdin /* input_stream */,
            post_dictionary_apache_key );

        if ( post_dictionary_datum() )
        {
            return this;
        }

    } /* If new attribute */

} /* while ( string_input() ) */

return null;

/* Attributes */
char *name;
POST_DICTIONARY_FILE *post_dictionary_file;
POST_DICTIONARY_RECALL *post_dictionary_recall;
char *datum;
```



POST_DICTIONARY_ATTRIBUTE (2)

```
/* Usage */
static char *post_dictionary_attribute_apache_marker(
    char *input )

/* Process */
char *piece(
    static apache_marker[],
    “”,
    input,
    0 );

/* Usage */
char *post_dictionary_attribute_name(
    char *input );

/* Process */
char *piece(
    attribute_name[],
    “”,
    input,
    1 );

/* Usage */
char *post_dictionary_attribute_datum(
    const char *WIDGET_SELECT_OPERATOR,
    FILE *input_stream,
    char *post_dictionary_apache_key );

/* Process */
char datum[];
char *ptr = datum;

while ( 1 )
{
    if ( !string_input(
            input[],
            input_stream,
            sizeof ( input )  ) )
    {
        break;
    }

    if ( !*input ) continue;

    if ( string_strncmp(
            input,
            “Content-Type: “ ) == 0 )
    {
        continue;
    }

    /* The last line will have “--” appended to the key */
    if ( string_strncmp(
            input,
            post_dictionary_apache_key ) == 0 )
    {
        break;
    }

    if ( *datum ) ptr += sprintf( ptr, “ “ );

    ptr += sprintf(
                ptr,
                “%s”,
                datum );
}

if ( strcmp(
        datum,
        WIDGET_SELECT_OPERATOR ) == 0 )
{
    return null;
}

char *strdup( string_trim( datum ) );
```

POST_DICTIONARY_FILE (1)

```
/* Usage */
POST_DICTIONARY_FILE *post_dictionary_file_new(
    const int POST_DICTIONARY_FILE_NAME_MAXSIZE,
    char *application_name,
    char *upload_directory,
    LIST *upload_filename_list,
    char *post_dictionary_apache_key,
    char *input,
    char *post_dictionary_attribute_name );

/* Process */
if ( !application_name ) return null;

boolean post_dictionary_file_boolean(
    upload_filename_list,
    post_dictionary_attribute_name );

if ( !post_dictionary_file_boolean() ) return null;

POST_DICTIONARY_FILE *post_dictionary_file_calloc(
    void );

char *post_dictionary_file_name_string(
    input );

if ( !post_dictionary_file_name_string() ) return this;

FILENAME *filename_new(
    post_dictionary_file_name_string(),
    POST_DICTIONARY_FILE_NAME_MAXSIZE
        /* probably 80 */,
    1 /* append_date_boolean */ );

char *post_dictionary_file_specification(
    application_name,
    upload_directory,
    filename_new()->return_string );

void post_dictionary_file_write(
    post_dictionary_apache_key,
    post_dictionary_file_specification() );

/* Attributes */
char *name_string;
FILENAME *filename;
char *specification;
```

input looks like:
Content-Disposition: form-data; name="exchange_format_filename"; filename="stmt(125).xf"

post_dictionary_attribute_name looks like:
exchange_format_filename

post_dictionary_file_name_string() looks like:
stmt(125).xf

FILENAME

```
char *clean
char *return_string
```

POST_DICTIONARY_FILE (2)

```
/* Usage */
char *post_dictionary_file_specification(
    char *application_name,
    char *upload_directory,
    char *filename_return_string );

/* Process */
snprintf(
    specification[],
    sizeof ( specification ),
    "%s/%s/%s",
    upload_directory,
    application_name,
    filename_return_string );

/* Usage */
boolean post_dictionary_file_boolean(
    LIST *upload_filename_list,
    char *post_dictionary_attribute_name );

/* Process */
boolean list_exists_string(
    post_dictionary_attribute_name,
    upload_filename_list );

if ( list_exists_string() )
{
    boolean file_boolean = 1;
}

/* Usage */
char *post_dictionary_file_name_string(
    char *input );

/* Process */
char *piece(
    name_string[],
    '"',
    input,
    3 );

if ( !*name_string ) return null;

char *string_trim( name_string );

/* Usage */
void post_dictionary_file_write(
    char *post_dictionary_apache_key,
    char *post_dictionary_file_specification );

/* Process */
char *apache_key_anchor =
    post_dictionary_apache_key;

int false_alarm_buffer_index = 0;

/* Skip "Content-Type: ..." */
void file_skip_line( stdin );

/* Skip "\n" */
void file_skip_line( stdin );

FILE *appaserver_output_file(
    post_dictionary_file_specification );

while ( ( int c = getc( stdin ) ) != EOF )
{
    if ( c == *apache_key )
    {
        apache_key++;

        /* All done */
        if ( !*apache_key ) break;

        false_alarm_buffer[
            false_alarm_buffer_index++ ] = c;

        continue;
    }

    /* If started an all done, but it's not all done. */
    if (    post_dictionary_apache_key !=
            apache_key_anchor )
    {
        for ( int i = 0; i < false_alarm_buffer_index; i++ )
        {
            putc(
                false_alarm_buffer[ i ],
                appaserver_output_file() );
        }

        post_dictionary_apache_key =
            apache_key_anchor;

        false_alarm_buffer_index = 0;

        if ( c == *post_dictionary_apache_key )
        {
            apache_key++;

            false_alarm_buffer[
                false_alarm_buffer_index++ ] = c;

            continue;
        }

        putc( c, appaserver_output_file() );
    }

fclose( appaserver_output_file() );
```

POST_DICTIONARY_RECALL

```
/* Usage */
POST_DICTIONARY_RECALL *post_dictionary_recall_new(
    char *WIDGET_UPLOAD_RECALL_PREFIX,
    const int POST_DICTIONARY_FILE_NAME_MAXSIZE,
    char *application_name,
    char *upload_directory,
    char *post_dictionary_apache_key,
    char *post_dictionary_attribute_name );

/* Process */
if ( !application_name ) return null;

boolean post_dictionary_recall_boolean(
    const char *WIDGET_UPLOAD_RECALL_PREFIX,
    char *post_dictionary_attribute_name );

if ( !post_dictionary_file_recall_boolean() ) return null;

POST_DICTIONARY_RECALL *post_dictionary_recall_calloc(
    void );

char *post_dictionary_attribute_datum(
    WIDGET_SELECT_OPERATOR,
    stdin /* input_stream */,
    post_dictionary_apache_key );

if ( !post_dictionary_attribute_datum() ) return this;

FILENAME *filename_new(
    post_dictionary_attribute_datum(),
    POST_DICTIONARY_FILE_NAME_MAX_SIZE
        /* probably 80 */,
    1 /* append_date_boolean */ );

char *post_dictionary_file_specification(
    application_name,
    upload_directory,
    filename_new()->return_string );

/* [Recall] may retrieve another application's filename */
boolean file_exists_boolean(
    post_dictionary_file_specification() );

if ( !file_exist_boolean() ) return this;

char *post_dictionary_recall_attribute_name(
    char *WIDGET_UPLOAD_RECALL_PREFIX,
    char *post_dictionary_attribute_name );

/* Attributes */
char *post_dictionary_attribute_datum;
FILENAME *filename;
char *post_dictionary_file_specification;
char *attribute_name;
```

post_dictionary_attribute_name looks like:
rrecall_exchange_format_filename

post_dictionary_attribute_datum() looks like:
stmt(125).xf

FILENAME

```
char *clean
char *return_string
```

FRAMESET (1)

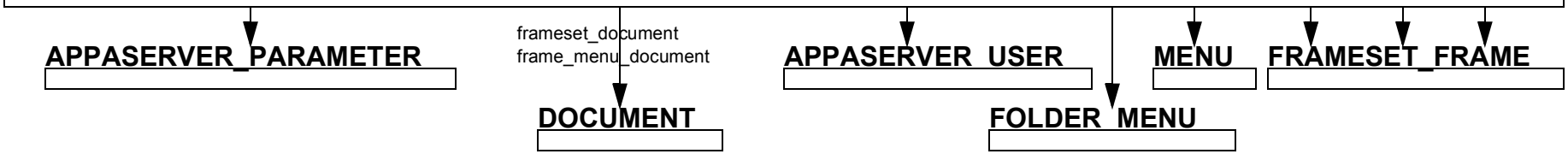
```
/* Usage */
FRAMESET *frameset_new(
    char *applicatation_name,
    char *session_key,
    char *login_name,
    boolean application_menu_horizontal_boolean(
        application_name ) );

/* Process */
APPASERVER_PARAMETER *appaserver_parameter =
    appaserver_parameter_new();

DOCUMENT *frameset_document =
    document_basic_new(
        application_name,
        (LIST *)0 /* javascript_filename_list */,
        application_title_string(
            application_name ) );

boolean entity_contact_key_boolean(
    ENTITY_TABLE,
    ENTITY_CONTACT_KEY_COLUMN );

APPASERVER_USER *appaserver_user_login_fetch(
    login_name,
    1 /* fetch_role_name_list */ );

char *login_default_role_name(
    LOGIN_DEFAULT_ROLE_TABLE,
    appaserver_user_login_fetch()->full_name,
    appaserver_user_login_fetch()->contact_key,
    entity_contact_key_boolean() );

char *frameset_role_name(
    LIST *appaserver_user_login_fetch()->
        role_appaserver_user_name_list,
    char *login_default_role_name() );

if ( frameset_role_name() )
{
    FOLDER_MENU *folder_menu =
        folder_menu_fetch(
            application_name,
            session_key,
            role_name,
            appaserver_parameter->data_directory,
            0 /* not folder_menu_remove_boolean */ );

    MENU *menu =
        menu_new(
            application_name,
            session_key,
            login_name,
            role_name,
            application_menu_horizontal_boolean,
            folder_menu->count_list );

    DOCUMENT *frame_menu_document =
        document_new(
            application_name,
            application_title_string( application_name ),
            (char *)0 /* title_html */,
            (char *)0 /* sub_title_html */,
            (char *)0/* sub_sub_title_html */,
            (char *)0 /* notepad */,
            (char *)0 /* onload_javascript_string */,
            menu->html,
            document_head_menu_setup_string(
                application_menu_horizontal_boolean ),
            (char *)0 /* calendar_setup_string */,
            javascript_include_string(
                application_name,
                (LIST *)0 /* javascript_filename_list */ ) );

    char *document_form_html(
        frame_menu_document->html,
        frame_menu_document->document_head->html,
        document_head_close_tag(),
        frame_menu_document->document_body->html,
        (char *)0 /* form_html */,
        document_body_close_tag(),
        document_close_tag() );
}

if ( !application_menu_horizontal_boolean )
{
    FRAMESET_FRAME *frameset_frame_menu =
        frameset_frame_new(
            application_name,
            appaserver_parameter->
                data_directory,
            session_key,
            FRAMESET_MENU_FRAME,
            document_form_html() );
        document_form_html() = (char *)0;
}

FRAMESET_FRAME *frameset_frame_prompt =
    frameset_frame_new(
        application_name,
        appaserver_parameter->
            data_directory,
        session_key,
        FRAMESET_PROMPT_FRAME,
        document_form_html() );

FRAMESET_FRAME *frameset_frame_table =
    frameset_frame_new(
        application_name,
        appaserver_parameter->
            data_directory,
        session_key,
        FRAMESET_TABLE_FRAME,
        (char *)0 /* document_form_html */ );

if ( !frameset_role_name() )
{
    char *frameset_filename(
        FRAMESET_FRAME *frameset_frame_menu,
        FRAMESET_FRAME *frameset_frame_prompt );

    char *execute_system_string_choose_role(
        CHOOSE_ROLE_EXECUTABLE,
        session_key,
        login_name,
        application_menu_horizontal_boolean,
        frameset_filename() );
}

char *blank_table_frame_system_string =
    char *frameset_blank_system_string(
        char *application_name,
        char *application_background_color(
            application_name ),
        (char *)0 /* application_title_string */,
        char *frameset_frame_table->
            output_filename,
        char *appaserver_error_filename(
            application_name ) );
```

FRAMESET (2)

```
/* Process (continued) */
if ( !application_menu_horizontal_boolean )
{
    char *blank_prompt_frame_system_string =
        char *frameset_blank_system_string(
            application_name,
            application_background_color(
                application_name ),
            application_title_string(
                application_name ),
            frameset_frame_prompt->
                output_filename,
            appaserver_error_filename(
                application_name ) );
}

char *frameset_html(
    char *frameset_document->html,
    char *frameset_document->document_head->html,
    char *document_head_close_tag(),
    FRAMESET_FRAME *frameset_frame_menu,
    char *frameset_frame_prompt->tag,
    char *frameset_frame_table->tag,
    char *document_close_tag() );

/* Usage */
```

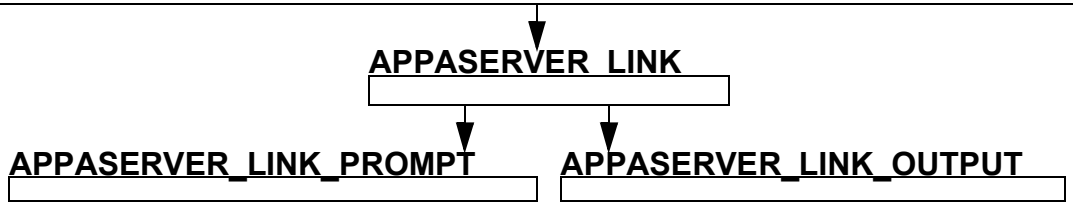
```
char *frameset_current_frame(
    const char *FRAMESET_PROMPT_FRAME,
    const char *FRAMESET_TABLE_FRAME,
    char *folder->appaserver_form );

/* Process */
boolean folder_form_prompt_boolean(
    FORM_APPASERVER_PROMPT,
    appaserver_form );

if ( folder_form_prompt_boolean() )
    return FRAMESET_PROMPT_FRAME;
else
    return FRAMESET_TABLE_FRAME;

/* Driver */
system( frameset->blank_table_frame_system_string );

if ( frameset->execute_system_string_choose_role )
{
    system(
        frameset->
            execute_system_string_choose_role );
}

if ( frameset->blank_prompt_frame_system_string )
```

```
{
    system(
        frameset->
            blank_prompt_frame_system_string );
}

printf( "%s\n", frameset->html );

/* Attributes */
APPASERVER_PARAMETER *appaserver_parameter;
DOCUMENT *frameset_document;
APPASERVER_USER *appaserver_user;
char *login_default_role_name;
char *role_name;
DOCUMENT *frame_menu_document;
FOLDER_MENU *folder_menu;
MENU *menu;
char *document_form_html;
FRAMESET_FRAME *frameset_frame_menu;
FRAMESET_FRAME *frameset_frame_prompt;
FRAMESET_FRAME *frameset_frame_table;
char *filename;
char *execute_system_string_choose_role;
char *blank_table_frame_system_string;
char *blank_prompt_frame_system_string;
char *html;
```



FRAMESET_FRAME

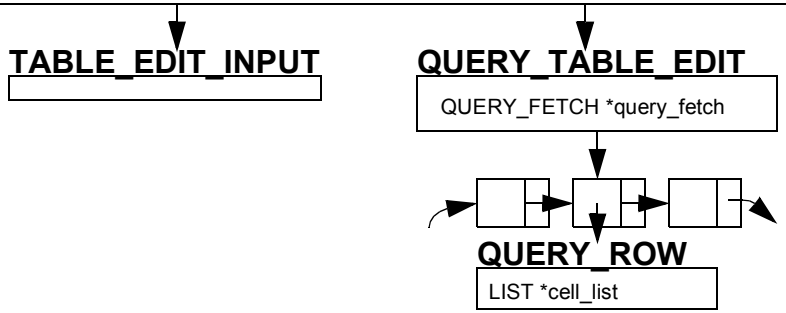
```
/* Usage */
FRAMESET_FRAME *frameset_frame_new(
    char *application_name,
    char *data_directory,
    char *session_key,
    char *frame_name,
    char *document_form_html );

/* Process */
APPASERVER_LINK *appaserver_link =
    appaserver_link_new(
        application_http_prefix(
            application_ssl_support_boolean(
                application_name ) ),
        application_server_address(),
        data_directory,
        frame_name /* filename_stem */,
        application_name,
        0 /* process_id */,
        session_key,
        (char *)0 /* begin_date_string */,
        (char *)0 /* end_date_string */,
        "html" /* extension */ );

char *prompt_filename =
    appaserver_link->
        appaserver_link_prompt->
        filename;

char *output_filename =
    appaserver_link->
        appaserver_link_output->
        filename;

if ( document_form_html )
{
    boolean string_file_write(
        output_filename,
        document_form_html /* string */ );

    if ( !string_file_write() ) exit( 1 );
}
else
{
    timlib_zap_file( output_filename );
}

char *frameset_frame_tag(
    frame_name,
    prompt_filename );

/* Usage */
char *frameset_frame_tag(
    char *frame_name,
    char *prompt_filename );

/* Process */
snprintf(
    tag[],
    sizeof ( tag ),
    "<frame name=\"%s\" src=\"%s\">",
    frame_name,
    prompt_filename );

/* Attributes */
APPASERVER_LINK *appaserver_link;
char *prompt_filename;
char *output_filename;
char *tag;
```



TABLE_EDIT (1)

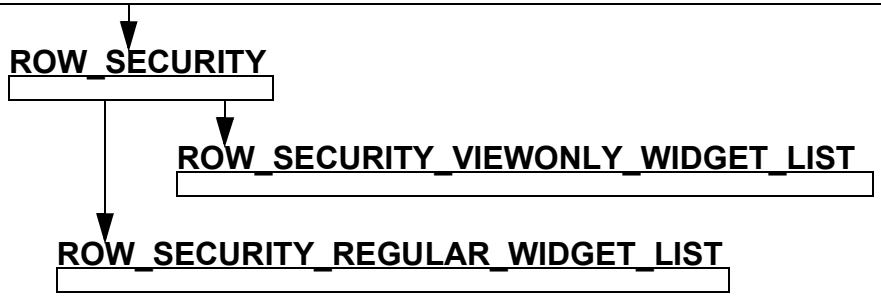
```
/* Usage */
TABLE_EDIT *table_edit_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *target_frame,
    pid_t process_id,
    LIST *relation_foreign_key_list,
    char *results_string,
    char *error_string,
    boolean application_menu_horizontal_boolean,
    DICTIONARY *post_dictionary->
        original_post_dictionary,
    char *appaserver_parameter->data_directory,
    char *drilldown_base_folder_name,
    char *drilldown_primary_data_list_string,
    LIST *drilldown_relation_foreign_key_list,
    boolean viewonly_boolean,
    boolean omit_delete_boolean );

/* Process */
TABLE_EDIT *table_edit_calloc(
    void );
```

```
TABLE_EDIT_INPUT *table_edit_input =
    table_edit_input_new(
        application_name,
        session_key,
        login_name,
        role_name,
        folder_name,
        target_frame,
        application_menu_horizontal_boolean,
        original_post_dictionary,
        data_directory,
        drilldown_base_folder_name,
        viewonly_boolean,
        omit_delete_boolean );

if ( !table_edit_input->table_edit_state ) return NULL;

QUERY_TABLE_EDIT *query_table_edit =
    /* Note: frees each where. */
    query_table_edit_new(
        application_name,
        login_name,
        folder_name,
        table_edit_input->security_entity_where,
        table_edit_input->
            dictionary_separate->
            no_display_name_list,
        table_edit_input->
            exclude_lookup_attribute_name_list,
        table_edit_input->relation_mto1_list,
        table_edit_input->relation_mto1_isa_list,
        table_edit_input->
            folder_attribute_append_isa_list,
        table_edit_input->
            folder->
            folder_attribute_primary_key_list,
        table_edit_input->
            folder->
            folder_attribute_name_list,
        table_edit_input->
            dictionary_separate->
            query_dictionary,
        table_edit_input->
            dictionary_separate->
            sort_dictionary,
        table_edit_input->row_security_role_update_list,
        TABLE_EDIT_QUERY_MAX_ROWS );

int table_edit_query_row_list_length(
    int query_table_edit->
        query_fetch->
        row_list );
```



TABLE_EDIT (2)

```
/* Process (continued) */
ROW_SECURITY *row_security =
    row_security_new(
        application_name,
        login_name,
        role_name,
        folder_name,
        table_edit_input->
            folder_attribute_non_primary_name_list,
        table_edit_input->relation_mto1_list,
        table_edit_input->relation_mto1_isa_list,
        table_edit_input->relation_one2m_join_list,
        table_edit_input->folder_attribute_append_isa_list,
        table_edit_input->folder->post_change_javascript,
        table_edit_input->
            folder_row_level_restriction->
            non_owner_viewonly,
        table_edit_input->
            dictionary_separate->
            drillthru_dictionary
            /* drop_down_dictionary */,
        table_edit_input->
            dictionary_separate->
            no_display_name_list,
        query_table_edit->query_select_name_list,
        table_edit_input->viewonly_attribute_name_list,
        table_edit_input->table_edit_state,
        table_edit_input->row_security_role_update_list,
        table_edit_query_row_list_length() );
```



TABLE_EDIT (3)

```
/* Process (continued) */
char *table_edit_post_action_string(
    POST_TABLE_EDIT_EXECUTABLE,
    application_name,
    session_key,
    login_name,
    role_name,
    folder_name,
    target_frame,
    process_id,
    drilldown_base_folder_name,
    drilldown_primary_data_list_string,
    drilldown_relation_foreign_key_list );

LIST *table_edit_heading_label_list(
        row_security->
            row_security_regular_widget_list,
        row_security->
            row_security_viewonly_widget_list );

LIST *table_edit_heading_name_list(
        row_security->
            row_security_regular_widget_list,
        row_security->
            row_security_viewonly_widget_list );

boolean table_edit_omit_heading_delete_checkbox_boolean(
    ROW_SECURITY_VIEWONLY_WIDGET_LIST *
        row_security->
            row_security_viewonly_widget_list );

FORM_TABLE_EDIT *form_table_edit =
    form_table_edit_new(
        folder_name,
        target_frame,
        process_id,
        table_edit_input->folder->post_change_javascript,
        table_edit_input->
            dictionary_separate->
            query_dictionary,
        table_edit_input->
            dictionary_separate->
            drillthru_dictionary,
        table_edit_input->
            dictionary_separate->
            no_display_dictionary,
        table_edit_input->folder_operation_list,
        table_edit_query_row_list_length(),
        table_edit_post_action_string(),
        table_edit_heading_label_list(),
        table_edit_heading_name_list(),
        table_edit_omit_heading_delete_checkbox_boolean() );

char *appaserver_update_filespecification(
    application_name,
    folder_name,
    session_key,
    process_id,
    relation_foreign_key_list,
    data_directory );

if ( !drilldown_base_folder_name )
{
    char *table_edit_document_notepad(
        const char *FRAMESET_PROMPT_FRAME,
        char *target_frame,
        char *table_edit_input->
            folder->
            notepad );

    TABLE_EDIT_DOCUMENT *
        table_edit_document =
            table_edit_document_new(
                application_name,
                session_key,
                login_name,
                role_name,
                folder_name,
                table_edit_input->table_edit_state,
                table_edit_input->
                    folder_attribute_append_isa_list,
                table_edit_input->
                    onload_javascript_string,
                results_string,
                error_string,
                table_edit_input->
                    folder->
                    javascript_filename,
                form_table_edit->sort_buttons_boolean,
                table_edit_input->
                    folder->
                    post_change_javascript,
                table_edit_input->menu,
                row_security,
                table_edit_document_notepad() );
}

if ( list_length(
        table_edit_input->
            relation_one2m_join_list ) )
{
    RELATION_JOIN *relation_join =
        relation_join_new(
            table_edit_input->
                folder->
                folder_attribute_primary_key_list
                /* one_folder_primary_key_list */,
            table_edit_input->
                relation_one2m_join_list,
            query_table_edit->
                query_fetch->
                row_list );
}

/* Attributes */
TABLE_EDIT_INPUT *table_edit_input;
QUERY_TABLE_EDIT *query_table_edit;
int query_row_list_length;
boolean viewonly_boolean;
ROW_SECURITY *row_security;
char *post_action_string;
LIST *heading_label_list;
LIST *heading_name_list;
boolean omit_heading_delete_checkbox_boolean;
FORM_TABLE_EDIT *form_table_edit;
char *appaserver_update_filespecification;
LIST *regular_widget_container_list;
LIST *viewonly_widget_container_list;
char *document_notepad;
TABLE_EDIT_DOCUMENT *document;
RELATION_JOIN *relation_join;
```

FORM_TABLE_EDIT

TABLE_EDIT_DOCUMENT

DOCUMENT *document

RELATION_JOIN

TABLE_EDIT (4)

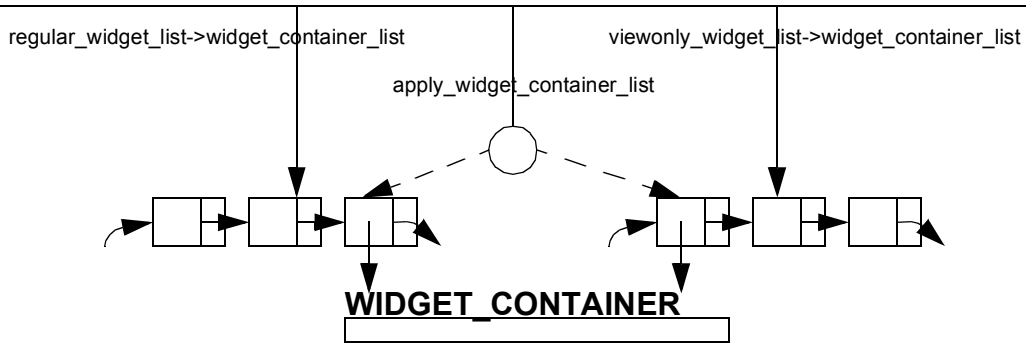
```
/* Driver */
void table_edit_spool_file_output(
    table_edit->appaserver_update_filespecification,
    table_edit->
        query_table_edit->
        query_fetch->
        row_list /* query_fetch_row_list */ );

void table_edit_output(
    APPASERVER_USER_LOGIN_NAME,
    login_name,
    table_edit->
        table_edit_input->
        folder_row_level_restriction->
        non_owner_viewonly,
    table_edit->
        table_edit_document->
        html,
    table_edit->form_table_edit->open_html,
    table_edit->
        form_table_edit->
        heading_container_string,
    table_edit->
        table_edit_input->
        folder_operation_list,
    table_edit->
        query_table_edit->
        query_fetch->
        row_list /* query_fetch_row_list */,
    table_edit->relation_join,
    table_edit->
        row_security->
        regular_widget_list,
    table_edit->
        row_security->
        viewonly_widget_list,
    table_edit->
        table_edit_input->
        row_security_role_update_list,
    table_edit->table_edit_input->table_edit_state,
    table_edit->form_table_edit->close_html,
    table_edit->
        table_edit_document->
        trailer_html );

/* Usage */
void table_edit_spool_file_output(
    char *appaserver_update_filespecification(),
    LIST *query_fetch_row_list );

/* Process */
FILE *table_edit_spool_file_open(
    char *appaserver_update_filespecification );

int row_number = 0;

for QUERY_ROW *query_row in query_fetch_row_list
{
    void table_edit_spool_file_cell_list_output(
        table_edit_spool_file_open(),
        ++row_number,
        query_row->cell_list );
}

void fclose( table_edit_spool_file_open() );

/* Usage */
void table_edit_spool_file_cell_list_output(
    FILE *table_edit_spool_file_open,
    int row_number,
    LIST *query_row->cell_list );

/* Process */
for QUERY_CELL *query_cell in cell_list
{
    if ( query_cell->folder_attribute )
    {
        char *string_escape_character(
            destination[],
            query_cell->select_datum,
            SQL_DELIMITER );

        fprintf(
            table_edit_spool_file_open,
            "%s_%d%c%s\n",
            query_cell->
                folder_attribute->
                attribute_name,
            row_number,
            SQL_DELIMITER,
            destination );
    }
}
```

TABLE_EDIT (5)

```
/* Driver */
void table_edit_output(
    const char *APPASERVER_USER_LOGIN_NAME,
    char *login_name,
    boolean table_edit->
        table_edit_input->
        folder_row_level_restriction->
        non_owner_viewonly,
    char *document_html,
    char *form_table_edit_open_html,
    char *form_table_edit_heading_container_string(),
    LIST *folder_operation_list,
    LIST *query_fetch_row_list,
    RELATION_JOIN *relation_join,
    ROW_SECURITY_REGULAR_WIDGET_LIST *
        regular_widget_list,
    ROW_SECURITY_VIEWONLY_WIDGET_LIST *
        viewonly_widget_list,
    ROW_SECURITY_ROLE_UPDATE_LIST *
        row_security_role_update_list,
    char *table_edit_state,
    char *form_table_edit_close_html,
    char *document_trailer_html );

/* Process */
if ( document_html )
{
    printf(
        "%s\n",
        document_html );
}

printf(
    "%s\n",
    form_table_edit_open_html );

if (    list_length( query_fetch_row_list >
        TABLE_EDIT_QUERY_MAX_ROWS )
{
    printf(
    "<h3>Warning: Max rows of %d exceeded.</h3>\n",
    TABLE_EDIT_QUERY_MAX_ROWS );
}

void table_edit_output_row_list(
    APPASERVER_USER_LOGIN_NAME,
    login_name,
    non_owner_viewonly,
    form_table_edit_heading_container_string,
    folder_operation_list,
    query_fetch_row_list,
    relation_join,
    regular_widget_list,
    viewonly_widget_list,
    row_security_role_update_list,
    table_edit_state );

printf(
    "%s\n",
    widget_table_close_tag() );

void table_edit_output_row_list_hidden(
    APPASERVER_USER_LOGIN_NAME,
    login_name,
    non_owner_viewonly,
    query_fetch_row_list,
     regular_widget_list,
    viewonly_widget_list,
    row_security_role_update_list );

printf("%s\n", form_table_edit_close_html );

if ( document_trailer_html )
{
    printf( "%s\n", document_trailer_html );
}
```

TABLE_EDIT (6)

```
/* Usage */
void table_edit_output_row_list(
    const char *APPASERVER_USER_LOGIN_NAME,
    char *login_name,
    boolean non_owner_viewonly,
    char *form_table_edit_heading_container_string(),
    LIST *folder_operation_list(),
    LIST *query_fetch_row_list,
    RELATION_JOIN *relation_join,
    ROW_SECURITY_REGULAR_WIDGET_LIST *
        regular_widget_list,
    ROW_SECURITY_VIEWONLY_WIDGET_LIST *
        viewonly_widget_list,
    /* Null if not participating */
    ROW_SECURITY_ROLE_UPDATE_LIST *
        row_security_role_update_list,
    char *table_edit_state,
    char *target_frame );

/* Process */
int row_number = 0;

if ( relation_join )
        list_rewind( relation_join->relation_join_row_list );

for QUERY_ROW *query_row in query_fetch_row_list
{
    if (    row_number ==
            TABLE_EDIT_QUERY_MAX_ROWS )
                break;

    boolean table_edit_row_viewonly_boolean(
        APPASERVER_USER_LOGIN_NAME,
        login_name,
        non_owner_viewonly,
        query_row->cell_list,
        query_row->viewonly_boolean,
        row_security_role_update_list );

LIST *table_edit_apply_widget_container_list(
    regular_widget_list,
    viewonly_widget_list,
    table_edit_row_viewonly_boolean() );

char *background_color = form_background_color();

if ( relation_join_row_list )
{
    RELATION_JOIN_ROW *relation_join_row =
        list_get(
            relation_join->
                relation_join_row_list );

    list_next( relation_join->relation_join_row_list );
}

if ( row_number && row_number % 10 == 0 )
{
    printf(
        "%s\n",
        form_table_edit_heading_container_string );
}

char *table_edit_row_html(
    folder_operation_list,
    table_edit_state,
    ++row_number,
    query_row->cell_list,
    relation_join_row,
    table_edit_apply_widget_container_list(),
    table_edit_row_viewonly_boolean(),
    background_color );

printf(
    "%s\n%s\n",
    widget_table_row_tag( background_color ),
        table_edit_row_html() );

    free( table_edit_row_html() );
}

/* Usage */
LIST *table_edit_apply_widget_container_list(
    ROW_SECURITY_REGULAR_WIDGET_LIST *
        regular_widget_list,
    ROW_SECURITY_VIEWONLY_WIDGET_LIST *
        viewonly_widget_list,
    boolean table_edit_row_viewonly_boolean() );

/* Process */
if ( !regular_widget_list && !viewonly_widget_list )
{
    exit( 1 );
}

if ( table_edit_row_viewonly_boolean )
{
    if ( !viewonly_widget_list ) exit( 1 );

    LIST *apply_widget_container_list =
        viewonly_widget_list->widget_container_list;
}
else
if ( regular_widget_list )
{
    LIST *apply_widget_container_list =
        regular_widget_list->widget_container_list;
}
else
{
    LIST *apply_widget_container_list =
        viewonly_widget_list->widget_container_list;
}
```

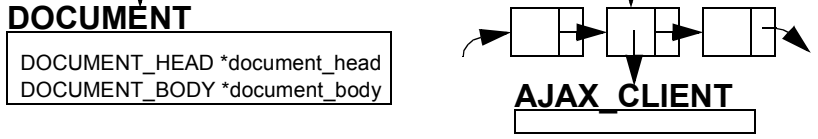


```
/* Usage */
char *table_edit_row_html(
    LIST *folder_operation_list(),
    char *table_edit_state,
    int row_number,
    LIST *query_row_cell_list,
    RELATION_JOIN_ROW *relation_join_row,
    LIST *table_edit_apply_widget_container_list,
    boolean table_edit_row_viewonly_boolean,
    char *form_background_color() );

/* Process */
char row_html[];
char *ptr = row_html;

for  FOLDER_OPERATION *folder_operation in
    folder_operation_list
{
    boolean table_edit_delete_mask_boolean(
        boolean folder_operation->
            operation->
            delete_boolean,
        boolean table_edit_row_viewonly_boolean );

    char *operation_html(
        folder_operation->
            operation->
            widget_container,
        table_edit_state,
        row_number,
        form_background_color,
        table_edit_delete_mask_boolean() );

    ptr +=
        sprintf(
            ptr,
            "%s%s\n",
            widget_table_data_tag(
                0 /* not align_right */,
                0 /* column_span */,
                0 /* width_em */,
                (char *)0 /* style */ ),
            operation_html() );

    free( operation_html() );
}

char *widget_table_edit_row_html(
    table_edit_state,
    row_number,
    query_row_cell_list,
    relation_join_row,
    table_edit_apply_widget_container_list,
    form_background_color,
    0 /* not hidden_only */ );

ptr += sprintf(
        ptr,
        "%s\n",
        widget_table_edit_row_html() );

free( widget_table_edit_row_html() );

/* Usage */
char *table_edit_state(
    char *folder_name,
    boolean viewonly_boolean,
    LIST *role_folder_list() );

/* Process */
if ( viewonly_boolean )
{
    return APPASERVER_VIEWONLY_STATE;
}

boolean role_folder_update_boolean(
    APPASERVER_UPDATE_STATE,
    folder_name,
    role_folder_list );

if ( role_folder_update_boolean() )
{
    return APPASERVER_UPDATE_STATE;
}

boolean role_folder_lookup_boolean(
    APPASERVER_LOOKUP_STATE,
    APPASERVER_UPDATE_STATE,
    folder_name,
    role_folder_list );

if ( role_folder_lookup_boolean() )
{
    return APPASERVER_VIEWONLY_STATE;
}

return NULL;

/* Usage */
LIST *table_edit_heading_label_list(
    ROW_SECURITY_REGULAR_WIDGET_LIST *
        row_security->
            row_security_regular_widget_list,
    ROW_SECURITY_VIEWONLY_WIDGET_LIST *
        row_security->
            row_security_viewonly_widget_list );

/* Process */
if ( row_security_regular_widget_list )
{
    LIST *widget_container_heading_label_list(
        row_security_regular_widget_list->
            widget_container_list );
}
else
{
    LIST *widget_container_heading_label_list(
        row_security_viewonly_widget_list->
            widget_container_list );
}

/* Usage */
LIST *table_edit_heading_name_list(
    ROW_SECURITY_REGULAR_WIDGET_LIST *
        row_security->
            row_security_regular_widget_list,
    ROW_SECURITY_VIEWONLY_WIDGET_LIST *
        row_security->
            row_security_viewonly_widget_list );

/* Process */
if ( row_security_regular_widget_list )
{
    LIST *widget_container_heading_name_list(
        row_security_regular_widget_list->
            widget_container_list );
}
else
{
    LIST *widget_container_heading_name_list(
        row_security_viewonly_widget_list->
            widget_container_list );
}
```

```
/* Usage */
void table_edit_output_row_list_hidden(
    const char *APPASERVER_USER_LOGIN_NAME,
    char *login_name,
    boolean non_owner_viewonly,
    LIST *folder_attribute_date_name_list,
    enum date_convert_format_enum
        date_convert_login_name_enum,
    LIST *query_fetch_row_list,
    ROW_SECURITY_REGULAR_WIDGET_LIST *
        regular_widget_list,
    ROW_SECURITY_VIEWONLY_WIDGET_LIST *
        viewonly_widget_list,
    /* Null if not participating */
    ROW_SECURITY_ROLE_UPDATE_LIST *
        row_security_role_update_list );

/* Process */
for QUERY_ROW *query_row in query_fetch_row_list
{
    boolean table_edit_row_viewonly_boolean(
        APPASERVER_USER_LOGIN_NAME,
        login_name,
        non_owner_viewonly,
        query_row->cell_list,
        query_row->viewonly_boolean,
        row_security_role_update_list );

    LIST *table_edit_apply_widget_container_list(
        regular_widget_list,
        viewonly_widget_list,
        table_edit_row_viewonly_boolean() );

    char *widget_table_edit_row_html(
        folder_attribute_date_name_list,
        date_convert_login_name_enum,
        (char *)0 /* table_edit_state */,
        ++row_number,
        query_row->cell_list,
        (RELATION_JOIN_ROW *)0,
        table_edit_apply_widget_container_list(),
        (char *)0 /* form_background_color */,
        1 /* hidden_only */ );

    printf(
        "%s\n",
        widget_table_edit_row_html() );

    free( widget_table_edit_row_html() );
}

/* Usage */
boolean table_edit_row_viewonly_boolean(
    const char *APPASERVER_USER_LOGIN_NAME,
    char *login_name,
    boolean non_owner_viewonly,
    LIST *query_row_cell_list,
    boolean query_row_viewonly_boolean,
    ROW_SECURITY_ROLE_UPDATE_LIST *
        row_security_role_update_list );

/* Process */
if ( query_row_viewonly_boolean ) return 1;

if ( non_owner_viewonly )
{
    /* Need to change to APPASERVER_USER compare */
    QUERY_CELL *query_cell =
        query_cell_seek(
            APPASERVER_USER_LOGIN_NAME
                /* attribute_name */,
            query_row_cell_list );

    if ( query_cell )
    {
        int string_strcmp(
            query_cell->select_datum,
            login_name );
        if ( string_strcmp() != 0 ) return 1;
    }
}

if ( row_security_role_update_list )
{
    QUERY_CELL *query_cell =
        query_cell_seek(
            row_security_role_update_list->
                attribute_not_null
                    /* attribute_name */,
            query_row_cell_list );

    if (    query_cell
    &&  query_cell->select_datum
    &&  *query_cell->select_datum )
    {
        return 1;
    }
}
return 0;

/* Usage */
char *table_edit_post_action_string(
    const char *POST_TABLE_EDIT_EXECUTABLE,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *target_frame,
    pid_t process_id,
    char *drilldown_base_folder_name,
    char *drilldown_primary_data_list_string,
    LIST *drilldown_relation_foreign_key_list );

/* Process */
```

TABLE_EDIT_DOCUMENT (1)

```
/* Usage */
TABLE_EDIT_DOCUMENT *
    table_edit_document_new(
        char *application_name,
        char *session_key,
        char *login_name,
        char *role_name,
        char *folder_name,
        char *table_edit_state,
        LIST *folder_attribute_append_isa_list,
        char *table_edit_input_onload_javascript_string,
        char *results_string,
        char *error_string,
        char *javascript_filename,
        boolean sort_buttons_boolean,
        char *post_change_javascript,
        MENU *menu,
        ROW_SECURITY *row_security,
        char *table_edit_document_notepad() );

/* Process */
TABLE_EDIT_DOCUMENT *
    table_edit_document_calloc(
        void );

char *table_edit_document_title_string(
    char *folder_name,
    char *table_edit_state );

int folder_attribute_calendar_date_name_list_length(
    folder_attribute_append_isa_list );

char *document_head_calendar_setup_string(
  folder_attribute_calendar_date_name_list_length());

if ( error_string && *error_string )
{
    results_string = NULL;
}

LIST *table_edit_document_javascript_filename_list(
    char *javascript_filename,
    boolean sort_buttons_boolean );

char *javascript_include_string(
    application_name,
    table_edit_document_javascript_filename_list() );

char *table_edit_document_javascript_replace(
    APPASERVER_UPDATE_STATE,
    table_edit_input_onload_javascript_string,
    post_change_javascript );

DOCUMENT *document =
    document_new(
        application_name,
        application_title_string( application_name ),
        table_edit_document_title_string(),
        results_string /* sub_title_string */,
        error_string /* sub_sub_title_string */,
        table_edit_document_notepad,
        table_edit_document_javascript_replace()
            /* onload_javascript_string */,
        (menu)
            ? menu->html : (char *)0,
        document_head_menu_setup_string(
            (menu)
            ? 1 : 0 /* menu_boolean */ ),
        document_head_calendar_setup_string(),
        javascript_include_string() );

LIST *row_security_ajax_client_list(
    row_security );

if ( list_length(
        row_security_ajax_client_list() ) )
{
    char *ajax_client_list_javascript(
        session_key,
        login_name,
        role_name,
        row_security_ajax_client_list() );
}

char *table_edit_document_html(
    char *document->html,
    char *document->document_head->html,
    char *document_head_close_tag(),
    char *document->document_body->html,
    char *ajax_client_list_javascript() );

char *table_edit_document_trailer_html(
    char *document_body_close_tag(),
    char *document_close_tag() );

/* Attributes */
char *title_string;
int folder_attribute_calendar_date_name_list_length;
char *document_head_calendar_setup_string;
LIST *javascript_filename_list;
char *javascript_include_string;
char *javascript_replace;
DOCUMENT *document;
LIST *row_security_ajax_client_list;
char *ajax_client_list_javascript;
char *html;
char *trailer_html;
```

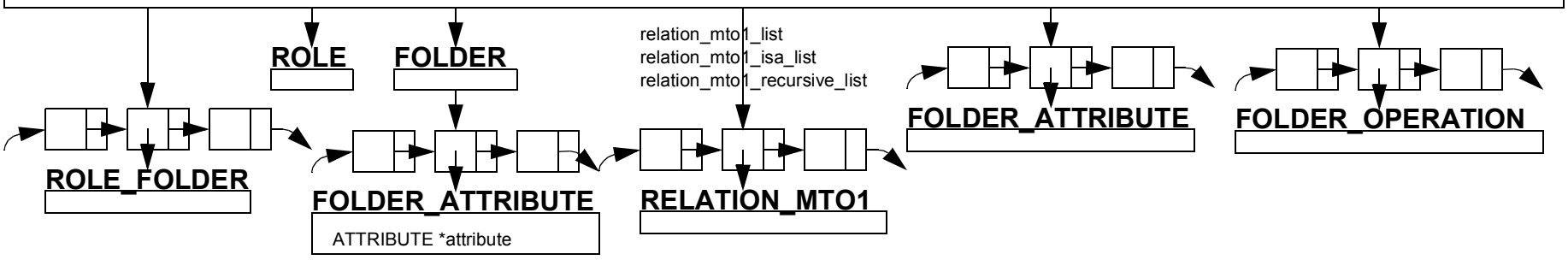


TABLE_EDIT_DOCUMENT (2)

```
/* Usage */
char *table_edit_document_javascript_replace(
    const char *APPASERVER_UPDATE_STATE,
    char *table_edit_input_onload_javascript_string,
    char *post_change_javascript );

/* Process */
char *document_javascript_replace[];
char *ptr = document_javascript_replace;

if ( post_change_javascript )
{
    char *javascript_replace(
        post_change_javascript,
        APPASERVER_UPDATE_STATE,
        JAVASCRIPT_ONLOAD_ROW_NUMBER
            /* probably -1 */ );

    ptr += sprintf( ptr,
        "%s",
        javascript_replace() );
}

if ( onload_javascript_string )
{
    if ( ptr != document_javascript_replace )
            ptr += sprintf( ptr, " && " );

    ptr += sprintf( ptr,
        "%s",
        onload_javascript_string );
}
```

TABLE_EDIT_INPUT (1)

```
/* Usage */
TABLE_EDIT_INPUT *table_edit_input_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *target_frame,
    boolean application_menu_horizontal_boolean,
    DICTIONARY *original_post_dictionary,
    char *data_directory,
    char *drilldown_base_folder_name,
    boolean viewonly_boolean,
    boolean omit_delete_boolean );

/* Process */
TABLE_EDIT_INPUT *table_edit_input_calloc(
    void );

LIST *role_folder_list(
    role_name,
    folder_name );

ROLE *role =
    role_fetch(
        role_name,
        1 /* fetch_role_attribute_exclude_list */ );

FOLDER *folder =
    folder_fetch(
        folder_name,
        exclude_lookup_attribute_name_list,
        /* Sets folder_attribute_primary_list */
        /* Sets folder_attribute_primary_key_list */
        /* Sets folder_attribute_name_list */
        1 /* fetch_folder_attribute_list */,
        1 /* fetch_attribute */,
        1 /* cache_boolean */ );

LIST *folder_attribute_non_primary_name_list(
    folder->folder_attribute_list );

LIST *exclude_lookup_attribute_name_list =
    role_attribute_exclude_name_list(
        APPASERVER_LOOKUP_STATE,
        role->role_attribute_exclude_list );

LIST *relation_mto1_list(
    folder_name /* many_folder_name */,
    folder->folder_attribute_primary_key_list
        /* many_primary_key_list */ );

LIST *relation_mto1_list() =
    relation_mto1_without_omit_drillthru_list(
        relation_mto1_list() );

LIST *relation_mto1_isa_list(
    (LIST *)0 /* mto1_isa_list Pass in null */,
    folder_name
        /* many_folder_name */,
    folder->folder_attribute_primary_key_list
        /* many_primary_key_list */,
    0 /* not fetch_relation_one2m_list */,
    1 /* fetch_relation_mto1_list */ );

LIST *relation_mto1_recursive_list(
    (LIST *)0 /* relation_mto1_list Pass in null */,
    folder_name /* many_folder_name */,
    folder->folder_attribute_primary_key_list
        /* many_primary_key_list */,
    0 /* not single_foreign_key_only */ );

void relation_mto1_list_set_one_to_many_list(
    relation_mto1_recursive_list()
        /* relation_mto1_list in/out */ );

LIST *relation_one2m_list(
    folder_name /* one_folder_name */,
    folder->folder_attribute_primary_key_list
        /* one_folder_primary_key_list */,
    0 /* not include_isa_boolean */ );

LIST *folder_attribute_append_isa_list(
    folder->folder_attribute_list,
    relation_mto1_isa_list() );

LIST *folder_operation_list(
     folder_name,
     role_name,
     1 /* fetch_operation */,
     1 /* fetch_process */ );

if ( omit_delete_boolean )
{
    folder_operation_list() =
        LIST *table_edit_input_trim_delete_operation_list(
            folder_operation_list() );
}
```



TABLE_EDIT_INPUT (2)

```
/* Process (continued) */
if ( list_length( relation_mto1_isa_list ) )
{
    void table_edit_input_isa_delete_prompt_set(
        folder_name,
        folder_operation_list() /* in/out */ );
}

char *table_edit_state(
    folder_name,
    viewonly_boolean,
    role_folder_list() );

if ( !table_edit_input_state() ) return this;

FOLDER_ROW_LEVEL_RESTRICTION *
    folder_row_level_restriction =
        folder_row_level_restriction_fetch(
            folder_name );

SECURITY_ENTITY *security_entity =
    security_entity_new(
        login_name,
        folder_name,
        folder_row_level_restriction->
            non_owner_forbid,
        role->override_row_restrictions );

LIST *folder_attribute_date_name_list(
    folder_attribute_append_isa_list() );

enum date_convert_format_enum
    date_convert_login_name_enum(
        application_name,
        login_name );

DICTIONARY_SEPARATE_TABLE_EDIT *
    dictionary_separate =
        dictionary_separate_table_edit_new(
            original_post_dictionary,
            application_name,
            login_name,
            folder_attribute_date_name_list(),
            folder_attribute_append_isa_list() );

char *table_edit_input_onload_javascript_string(
    const char *TABLE_EDIT_ONLOAD_JAVASCRIPT_KEY,
    DICTIONARY *dictionary_separate->
        non_prefixed_dictionary );

LIST *relation_one2m_join_list(
    folder_name,
    folder->folder_attribute_primary_key_list,
    dictionary_separate->no_display_dictionary );
```

relation_one2m_list
relation_one2m_join_list

SECURITY_ENTITY

FOLDER_ROW_LEVEL_RESTRICTION

boolean non_owner_viewonly
boolean non_owner_forbid

DICTIONARY_SEPARATE_TABLE_EDIT

DICTIONARY *sort_dictionary
DICTIONARY *drillthru_dictionary
DICTIONARY *no_display_dictionary
LIST *no_display_name_list
DICTIONARY *non_prefixed_dictionary
DICTIONARY *query_dictionary

RELATION_ONE2M

TABLE_EDIT_INPUT (3)

```
/* Process (continued) */
if ( !drilldown_base_folder_name )
{
    boolean menu_horizontal_boolean(
        target_frame /* current_frame */,
        FRAMESET_PROMPT_FRAME,
        application_menu_horizontal_boolean );
}

if ( menu_horizontal_boolean() )
{
    FOLDER_MENU *folder_menu =
        folder_menu_fetch(
            application_name,
            session_key,
            role_name,
            data_directory,
            0 /* not folder_menu_remove_boolean */ );

    MENU *menu =
        menu_new(
            application_name,
            session_key,
            login_name,
            role_name,
            1 /* application_menu_horizontal_boolean */,
            folder_menu->count_list );
}

if ( strcmp(
        table_edit_state(),
        APPASERVER_LOOKUP_STATE ) != 0 )
{
    ROW_SECURITY_ROLE_UPDATE_LIST *
        row_security_role_update_list =
            /* Returns null if not participating */
            row_security_role_update_list_fetch(
                role_name,
                folder_name,
                role->override_row_restrictions );
}

if ( !viewonly_boolean )
{
    DRILLTHRU_STATUS *drillthru_status =
        drillthru_status_new(
            dictionary_separate->drillthru_dictionary );

    int table_edit_input_max_foreign_key_list_length(
        boolean drillthru_status->skipped_boolean,
        char *drilldown_base_folder_name );

    LIST *table_edit_input_viewonly_attribute_name_list(
        role->role_attribute_exclude_list,
        relation_mto1_list(),
        folder_attribute_append_isa_list(),
        table_edit_input_max_foreign_key_list_length(),
        drillthru_status->skipped_boolean  );
}

enum date_convert_format_enum
    date_convert_login_name_enum(
        application_name,
        login_name );

/* Attributes */
LIST *role_folder_list;
ROLE *role;
LIST *exclude_lookup_attribute_name_list;
FOLDER *folder;
LIST *folder_attribute_non_primary_name_list;
LIST *relation_mto1_list;
LIST *relation_mto1_isa_list;
LIST *relation_mto1_recursive_list;
LIST *relation_one2m_list;
LIST *folder_attribute_append_isa_list;
LIST *folder_operation_list;
char *table_edit_state;
FOLDER_ROW_LEVEL_RESTRICTION *
    folder_row_level_restriction;
SECURITY_ENTITY *security_entity;
LIST *folder_attribute_date_name_list;
DICTIONARY_SEPARATE_TABLE_EDIT *
    dictionary_separate;
char *onload_javascript_string;
LIST *relation_one2m_join_list;
boolean menu_horizontal_boolean;
FOLDER_MENU *folder_menu;
MENU *menu;
ROW_SECURITY_ROLE_UPDATE_LIST *
    row_security_role_update_list;
int max_foreign_key_list_length;
LIST *viewonly_attribute_name_list;
enum date_convert_format_enum
    date_convert_login_name_enum;
```

TABLE_EDIT_INPUT (4)

```
/* Usage */
LIST *table_edit_input_trim_delete_operation_list(
    LIST *folder_operation_list() );

/* Process */
LIST *list = list_new();

for FOLDER_OPERATION *folder_operation in
    folder_operation_list
{
    if ( string_strncmp(
            folder_operation->operation_name,
            “delete” ) == 0 )
    {
        continue;
    }

    list_set(
        list,
        folder_operation );
}
```

```
/* Usage */
void table_edit_input_isa_delete_prompt_set(
    char *folder_name,
    LIST *folder_operation_list() /* in/out */ );

/* Process */
char *table_edit_input_isa_delete_prompt(
    char *folder_name );

for FOLDER_OPERATION *folder_operation in
    folder_operation_list
{
    if ( string_strcmp(
            folder_operation->operation_name,
            “delete_isa_only” ) == 0 )
    {
        folder_operation->
            operation->
            widget_container->
            checkbox->
            prompt =
                table_edit_input_isa_delete_prompt();

        folder_operation->
            operation->
            widget_container->
            heading_string =
                table_edit_input_isa_delete_prompt();
    }
}
```

TABLE_EDIT_INPUT (5)

```
/* Usage */
LIST *table_edit_input_viewonly_attribute_name_list(
    LIST *role_attribute_exclude_list,
    LIST *relation_mto1_list,
    LIST *folder_attribute_append_isa_list,
    int table_edit_input_max_foreign_key_list_length,
    boolean drillthru_status_skipped_boolean );

/* Process */
LIST *viewonly_attribute_name_list =
    role_attribute_exclude_name_list(
        APPASERVER_UPDATE_STATE,
        role_attribute_exclude_list );

if ( !viewonly_attribute_name_list )
{
    viewonly_attribute_name_list = list_new();
}

if (    list_length( relation_mto1_list )
&&  table_edit_input_max_foreign_key_list_length )
{
    LIST *relation_mto1_foreign_key_less_equal_list(
        QUERY_DROP_DOWN_FETCH_MAX_ROWS,
        max_foreign_key_list_length
         /* DRILLTHRU_SKIPPED_MAX_FOREIGN_LENGTH */,
        relation_mto1_list );

    for RELATION_MTO1 *relation_mto1 in
        relation_mto1_list
    {
        boolean table_edit_input_relation_not_used(
            relation_mto1,
            relation_mto1_foreign_key_less_equal_list() );
        if ( table_edit_input_relation_not_used() )
        {
            list_set_list(
                viewonly_attribute_name_list,
                relation_mto1->relation_foreign_key_list );
        }
    }
}

if (    drillthru_status_skipped_boolean
&&  list_rewind( relation_mto1_list ) )
{
    for RELATION_MTO1 *relation_mto1 in
        relation_mto1_list
    {
        if ( relation_mto1->one_folder->drillthru )
        {
            list_set_list(
                viewonly_attribute_name_list,
                relation_mto1->relation_foreign_key_list );
        }
    }
}

LIST *folder_attribute_viewonly_attribute_name_list(
    folder_attribute_append_isa_list );

list_set_list(
    viewonly_attribute_name_list,
    folder_attribute_viewonly_attribute_name_list() );

if ( !list_length( viewonly_attribute_name_list ) )
{
    list_free( viewonly_attribute_name_list );
    viewonly_attribute_name_list = (LIST *)0;
}

/* Usage */
boolean table_edit_input_relation_not_used(
    RELATION_MTO1 *relation_mto1,
    LIST *relation_mto1_foreign_key_less_equal_list );

/* Process */
for RELATION_MTO1 *relation_mto1_foreign in
    relation_mto1_foreign_key_less_equal_list
{
    if ( relation_mto1_foreign == relation_mto1 )
    {
        return 0;
    }
}
return 1;

/* Usage */
int table_edit_input_max_foreign_key_list_length(
    boolean drillthru_status_skipped_boolean,
    char *drilldown_base_folder_name );

/* Process */
if ( drillthru_status_skipped_boolean )
    max_foreign_key_list_length = 1;
else
if ( drilldown_base_folder_name )
    max_foreign_key_list_length = 2;
else
    max_foreign_key_list_length = 0;
```

POST_TABLE_EDIT (1)

```
/* Usage */
POST_TABLE_EDIT *post_table_edit_new(
    int argc,
    char **argv,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *target_frame,
    pid_t process_id,
    LIST *drilldown_relation_foreign_key_list );

/* Process */
POST_TABLE_EDIT *post_table_edit_calloc(
    void );

SESSION_FOLDER *session_folder =
    /* Sets appaserver environment and outputs argv */
    /* Each parameter is security inspected */
    session_folder_integrity_exit(
        argc,
        argv,
        application_name,
        session_key,
        login_name,
        role_name,
        folder_name,
        APPASERVER_LOOKUP_STATE,
        APPASERVER_UPDATE_STATE );

POST_TABLE_EDIT_INPUT *post_table_edit_input =
    post_table_edit_input_new(
        application_name,
        session_key,
        login_name,
        role_name,
        folder_name,
        target_frame,
        process_id,
        drilldown_relation_foreign_key_list );

if ( !post_table_edit_input ) return this;

if ( post_table_edit_input->
        post_table_edit_forbid() )
{
    exit( 1 );
}

if (    post_table_edit_input->
            role_folder_update_boolean
&&    dictionary_length(
            dictionary_separate->multi_row_dictionary ) )
{
    UPDATE *update =
        update_new(
            application_name,
            session_key,
            login_name,
            role_name,
            folder_name,
            post_table_edit_input->
                role->
                override_row_restrictions,
            post_table_edit_input->
                dictionary_separate->
                multi_row_dictionary,
            post_table_edit_input->
                file_dictionary,
            post_table_edit_input->
                folder_row_level_restriction,
            post_table_edit_input->
                relation_one2m_list,
            post_table_edit_input->
                relation_mto1_list,
            post_table_edit_input->
                relation_mto1_isa_list,
            post_table_edit_input->
                folder_attribute_append_isa_list,
            post_table_edit_input->
                post_change_process,
            1 /* update_root_boolean */ );
}
```

SESSION_FOLDER **POST_TABLE_EDIT_INPUT** **UPDATE**

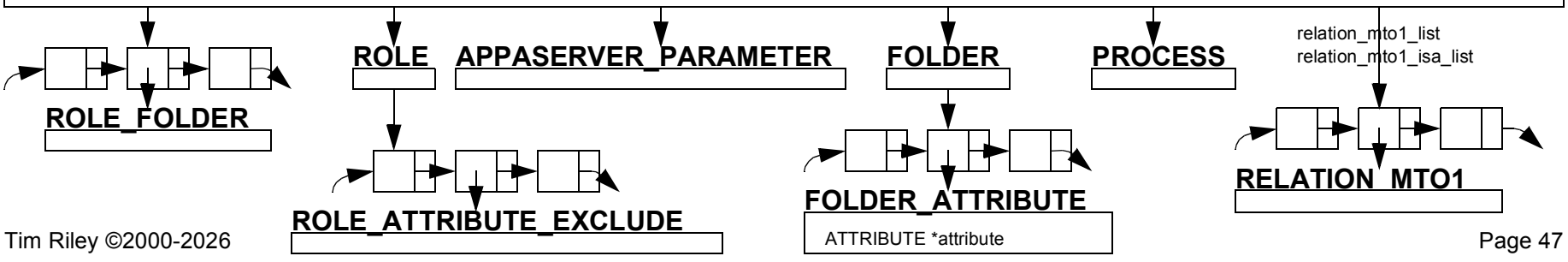
POST_TABLE_EDIT (2)

```
/* Process (continued) */
if ( dictionary_length(
        post_table_edit_input->
           dictionary_separate->
           operation_dictionary ) )
{
    OPERATION_ROW_LIST *operation_row_list =
        operation_row_list_new(
           application_name,
           session_key,
           login_name,
           role_name,
           folder_name,
           target_frame,
           post_table_edit_input->
              table_edit_state,
           post_table_edit_input->
              folder_operation_list,
           post_table_edit_input->
              folder->
              folder_attribute_primary_key_list,
           post_table_edit_input->
              folder->
              folder_attribute_name_list,
           post_table_edit_input->
              dictionary_separate->
              operation_dictionary,
           post_table_edit_input->
              dictionary_separate->
              no_display_dictionary,
           post_table_edit_input->
              dictionary_separate->
              multi_row_dictionary,
           post_table_edit_input->
              dictionary_separate->
              drillthru_dictionary,
           (update) ? update->results_string : (char *)0,
           (update) ? update->error_string : (char *)0 );
}

DICTIONARY *dictionary_separate_send_dictionary(
    post_table_edit_input->
       dictionary_separate->
       sort_dictionary,
    DICTIONARY_SEPARATE_SORT_PREFIX,
    post_table_edit_input->
       dictionary_separate->
       query_dictionary,
    DICTIONARY_SEPARATE_QUERY_PREFIX,
    post_table_edit_input->
       dictionary_separate->
       drillthru_dictionary,
    DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
    (DICTIONARY *)0 /* ignore_dictionary */,
    DICTIONARY_SEPARATE_IGNORE_PREFIX,
    post_table_edit_input->
       dictionary_separate->
       no_display_dictionary,
    DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
    post_table_edit_input->
       dictionary_separate->
       pair_dictionary,
    DICTIONARY_SEPARATE_PAIR_PREFIX,
    post_table_edit_input->
       dictionary_separate->
       prompt_dictionary
          /* non_prefixed_dictionary */ );

void dictionary_set(
    dictionary_separate_send_dictionary(),
    POST_TABLE_EDIT_FOLDER_LABEL,
    folder_name );

char *dictionary_separate_send_string(
    dictionary_separate_send_dictionary() );

/* Attributes */
pid_t process_id;
SESSION_FOLDER *session_folder;
POST_TABLE_EDIT_INPUT *post_table_edit_input;
UPDATE *update;
OPERATION_ROW_LIST *operation_row_list;
DICTIONARY *dictionary_separate_send_dictionary;
char *dictionary_separate_send_string;
```

```
/* Usage */
boolean post_table_edit_forbid(
    boolean role_folder_lookup_boolean );

/* Process */
1 - role_folder_lookup_boolean;

/* Driver */
if ( !post_table_edit->post_table_edit_input )
{
    document_process_output(
        application_name,
        (LIST *)0 /* javascript_filename_list */,
        "Expired Form" /* title_string */ );

    printf( "%s\n", POST_TABLE_EDIT_FORM_EXPIRED );
    document_close();
    exit( 0 );
}

if ( post_table_edit->update )
{
    char *update->error_string =
        char *update_row_list_execute(
            SQL_EXECUTABLE,
            application_name,
            1 /* update_root_boolean */,
            post_table_edit->
                update->
                update_row_list_distinct_sql_list,
            post_table_edit->
                post_table_edit_input->
                appaserver_error_filename );

     if ( !update->error_string )
     {
          void update_row_list_command_line_execute(
               post_table_edit->
                    update->
                    update_row_list );
     }
}

if (    !update->error_string
&&   post_table_edit->operation_row_list )
{
    char *operation_row_list_output_string =
        char *operation_row_list_execute(
            post_table_edit->operation_row_list );

    if ( post_table_edit->
            operation_row_list->
            output_boolean )
    {
        exit( 0 );
    }
}

if ( !operation_row_list_output_string )
{
    if ( *drilldown_base_folder_name )
    {
        char *string_unescape_character(
            drilldown_primary_data_list_unescape[]
                /* destination */,
            drilldown_primary_data_list_string
                /* datum */,
            ATTRIBUTE_MULTI_KEY_DELIMITER );

        char *system_string =
            char *execute_system_string_drilldown(
                POST_TABLE_EDIT_DRILLDOWN_EXECUTABLE,
                session_key,
                login_name,
                role_name,
                target_frame,
                drilldown_base_folder_name,
                drilldown_primary_data_list_unescape,
                update->results_string,
                update->error_string,
                post_table_edit->
                    dictionary_separate_send_string,
                post_table_edit->
                    post_table_edit_input->
                    appaserver_error_filename,
                getpid() /* parent_process_id */,
                1 /* operation_row_checked_count */ );
    }
    else
    {
        char *system_string =
            char *execute_system_string_table_edit(
                TABLE_EDIT_EXECUTABLE,
                session_key,
                login_name,
                role_name,
                folder_name,
                target_frame,
                update->results_string,
                update->error_string,
                post_table_edit->
                    dictionary_separate_send_string,
                post_table_edit->
                    post_table_edit_input->
                    appaserver_error_filename );
    }
}

if ( operation_row_list_output_string )
{
    printf( "%s\n", operation_row_list_output_string );
}
else
{
    security_system(
        SECURITY_FORK_CHARACTER,
        SECURITY_FORK_STRING,
        system_string );
}
```

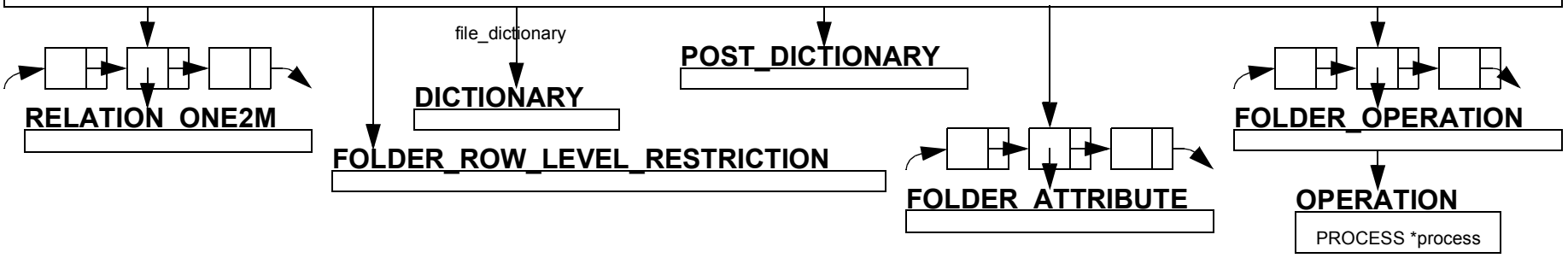
POST_TABLE_EDIT_INPUT (1)

```
/* Usage */
POST_TABLE_EDIT_INPUT *
    post_table_edit_input_new(
        char *application_name,
        char *session_key,
        char *login_name,
        char *role_name,
        char *folder_name,
        char *target_frame,
        pid_t process_id,
        LIST *drilldown_relation_foreign_key_list );

/* Process */
POST_TABLE_EDIT_INPUT *
    post_table_edit_input_calloc(
        void );

LIST *role_folder_list(
    role_name,
    folder_name );

boolean role_folder_lookup_boolean(
    APPASERVER_LOOKUP_STATE,
    APPASERVER_UPDATE_STATE,
    folder_name,
    role_folder_list() );

boolean post_table_edit_forbid(
    role_folder_lookup_boolean() );

if ( post_table_edit_forbid() ) return this;

boolean role_folder_update_boolean =
    role_folder_state_boolean(
        APPASERVER_UPDATE_STATE,
        folder_name,
        role_folder_list() );

char *table_edit_state(
    folder_name,
    0 /* not viewonly_boolean */,
    role_folder_list() );

ROLE *role =
    role_fetch(
        role_name,
        1 /* fetch_role_attribute_exclude_list */ );

LIST *exclude_attribute_name_list = list_new();

list_set_list(
    exclude_attribute_name_list,
    role_attribute_exclude_name_list(
        APPASERVER_UPDATE_STATE,
        role->role_attribute_exclude_list ) );

list_set_list(
    exclude_attribute_name_list,
    role_attribute_exclude_name_list(
        APPASERVER_LOOKUP_STATE,
        role->role_attribute_exclude_list ) );

APPASERVER_PARAMETER *
    appaserver_parameter =
        appaserver_parameter_new();

FOLDER *folder =
    folder_fetch(
        folder_name,
        /* Set to cache all folders for a role */
        role_name,
        exclude_attribute_name_list,
        /* Sets folder_attribute_primary_list */
        /* Sets folder_attribute_primary_key_list */
        /* Sets folder_attribute_name_list */
        1 /* fetch_folder_attribute_list */,
        1 /* fetch_attribute */ );

LIST *relation_mto1_isa_list(
    (LIST *)0 /* mto1_isa_list Pass in null */,
    folder_name
        /* many_folder_name */,
    folder->folder_attribute_primary_key_list
        /* many_folder_primary_key_list */,
    0 /* not fetch_relation_one2m_list */,
    0 /* not fetch_relation_mto1_list */ );

if ( role_folder_update_boolean )
{
    if ( folder->post_change_process_name )
    {
        PROCESS *post_change_process =
            process_fetch(
                folder->post_change_process_name,
                (char *)0 /* document_root */,
                (char *)0 /* relative_source */,
                1 /* check_executable_inside_filesystem */,
                appaserver_parameter->mount_point );
    }

    LIST *relation_mto1_list(
        folder_name
            /* many_folder_name */,
        folder->folder_attribute_primary_key_list
            /* many_folder_primary_key_list */ );
```



POST_TABLE_EDIT_INPUT (2)

```
/* Process (continued) */
   LIST *relation_one2m_list(
      folder_name /* one_folder_name */,
      folder->folder_attribute_primary_key_list
         /* one_folder_primary_key_list */,
      1 /* include_isa_boolean */ );

   FOLDER_ROW_LEVEL_RESTRICTION *
      folder_row_level_restriction =
         folder_row_level_restriction_fetch(
            folder_name );

   char *appaserver_update_filespecification(
      application_name,
      folder_name,
      session_key,
      process_id,
      drilldown_relation_foreign_key_list,
      appaserver_parameter->data_directory );

   DICTIONARY *file_dictionary =
      dictionary_file_fetch(
         appaserver_update_filespecification(),
         SQL_DELIMITER );

   if ( !file_dictionary ) return NULL;
}

LIST *folder_operation_list(
   folder_name,
   role_name,
   1 /* fetch_operation */,
   1 /* fetch_process */ );

LIST *folder_operation_name_list(
   folder_operation_list() );

POST_DICTIONARY *post_dictionary =
   post_dictionary_stdin_new(
      (char *)0 /* WIDGET_UPLOAD_RECALL_PREFIX */,
      (char *)0 /* application_name */,
      (char *)0 /* upload_directory */,
      (LIST *)0 /* upload_filename_list */ );

LIST *folder_attribute_append_isa_list(
   folder->folder_attribute_list /* append_isa_list */,
   relation_mto1_isa_list() );

LIST *folder_attribute_name_list(
   (char *)0 /* folder_name */,
   folder_attribute_append_isa_list() );

LIST *folder_attribute_date_name_list(
   folder_attribute_append_isa_list() );
```



POST_TABLE_EDIT_INPUT (3)

```
/* Process (continued) */
DICTIONARY_SEPARATE_POST_TABLE_EDIT *
    dictionary_separate =
        dictionary_separate_post_table_edit_new(
            post_dictionary->original_post_dictionary,
            application_name,
            login_name,
            relation_mto1_list(),
            folder_attribute_name_list(),
            folder_operation_name_list(),
            folder_attribute_date_name_list(),
            folder_attribute_append_isa_list() );

void post_table_edit_input_dictionary_set_time(
    dictionary_separate->
        multi_row_dictionary /* in/out */,
    folder_attribute_append_isa_list() );

char *appaserver_error_filespecification(
    application_name );

/* Usage */
void post_table_edit_input_dictionary_set_time(
    DICTIONARY *multi_row_dictionary /* in/out */,
    LIST *folder_attribute_append_isa_list );

/* Process */
LIST *dictionary_key_list( multi_row_dictionary );

for char *key in dictionary_key_list()
{
    dictionary_get( key, multi_row_dictionary );

    if (    !dictionary_get()
    ||      !*dictionary_get()
    ||      *dictionary_get() == '/' )
    {
        continue;
    }

    (void)string_trim_index( trim_index[], key );

    FOLDER_ATTRIBUTE *folder_attribute_seek(
        (char *)0 /* folder_name */,
        trim_index,
        folder_attribute_append_isa_list );

    if ( !folder_attribute_seek() ) continue;

    boolean attribute_is_date_time(
        folder_attribute_seek()->
            attribute->
            datatype_name );

    if ( !attribute_is_date_time() ) continue;

    boolean string_character_boolean(
        dictionary_get() /* datum */,
        ' ' /* c */ );

    if ( string_character_boolean() ) continue;

    char *date_set_now_hhmmss(
        dictionary_get() /* date_string */,
        date_utc_offset() );

    if ( date_set_now_hhmmss() )
    {
        dictionary_set(
            multi_row_dictionary,
            key,
            date_set_now_hhmmss() );
    }

    /* Attributes */
    LIST *role_folder_list;
    boolean role_folder_lookup_boolean;
    boolean post_table_edit_forbid;
    boolean role_folder_update_boolean;
    char *table_edit_state;
    ROLE *role;
    LIST *exclude_attribute_name_list;
    APPASERVER_PARAMETER *appaserver_parameter;
    FOLDER *folder;
    LIST *relation_mto1_isa_list;
    PROCESS *post_change_process;
    LIST *relation_mto1_list;
    LIST *relation_one2m_list;
    FOLDER_ROW_LEVEL_RESTRICTION *
        folder_row_level_restriction;
    char *appaserver_update_filespecification;
    DICTIONARY *file_dictionary;
    LIST *folder_operation_list;
    LIST *folder_operation_name_list;
    POST_DICTIONARY *post_dictionary;
    LIST *folder_attribute_append_isa_list;
    LIST *folder_attribute_name_list;
    LIST *folder_attribute_date_name_list;
    DICTIONARY_SEPARATE_POST_TABLE_EDIT *
        dictionary_separate;
    char *appaserver_error_filespecification;
```

DICTIONARY_SEPARATE_POST_TABLE_EDIT

```
DICTIONARY *sort_dictionary
DICTIONARY *query_dictionary
DICTIONARY *drillthru_dictionary
DICTIONARY *no_display_dictionary
DICTIONARY *pair_dictionary
DICTIONARY *non_prefixed_dictionary
DICTIONARY *multi_row_dictionary
DICTIONARY *prompt_dictionary
DICTIONARY *operation_dictionary
LIST *no_display_name_list
```

TABLE_INSERT (1)

```
/* Usage */
TABLE_INSERT *table_insert_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *target_frame,
    char *result_string,
    char *error_string,
    DICTIONARY *post_dictionary->
        original_post_dictionary );

/* Process */
TABLE_INSERT *table_insert_calloc( void );

TABLE_INSERT_INPUT *
    table_insert_input =
        table_insert_input_new(
            application_name,
            session_key,
            login_name,
            role_name,
            folder_name,
            target_frame,
            original_post_dictionary );

if ( table_insert_input->table_insert_forbid ) exit( 1 );

boolean table_insert_entire_primary_key_ignored(
    table_insert_input->
        folder->
        folder_attribute_primary_key_list,
    table_insert_input->
        dictionary_separate->
        ignore_name_list );

if ( table_insert_entire_primary_key_ignored() )
    return this;

VERTICAL_NEW_TABLE *vertical_new_table =
    vertical_new_table_new(
        VERTICAL_NEW_PROMPT_ONE_HIDDEN_LABEL,
        VERTICAL_NEW_PROMPT_MANY_HIDDEN_LABEL,
        FRAMESET_PROMPT_FRAME,
        application_name,
        session_key,
        login_name,
        role_name,
        table_insert_input->
            appaserver_parameter_data_directory,
        table_insert_input->
            dictionary_separate->
            non_prefixed_dictionary,
        table_insert_input->
            menu_horizontal_boolean );

if ( table_insert_input->
        relation_mto1_automatic_preselection )
{
    QUERY_DICTIONARY *query_dictionary =
        query_dictionary_fetch(
            application_name,
            table_insert_input->
                dictionary_separate->
                prompt_dictionary,
            table_insert_input->
                relation_mto1_automatic_preselection->
                many_folder_name,
            table_insert_input->
                relation_mto1_automatic_preselection->
                relation_foreign_key_list
                    /* select_attribute_name_list */,
            table_insert_input->
                pair_one2m_table_insert->
                folder_attribute_primary_key_list
                    /* where_attribute_name_list */ );

FORM_TABLE_INSERT_AUTOMATIC *
    form_table_insert_automatic =
        form_table_insert_automatic_new(
            application_name,
            session_key,
            login_name,
            role_name,
            folder_name,
            target_frame,
            table_insert_input->
                folder->
                post_change_javascript,
            table_insert_input->relation_mto1_list
                /* many_folder_relation_mto1_list */,
            table_insert_input->relation_mto1_isa_list,
            table_insert_input->
                folder_attribute_append_isa_list,
            table_insert_input->
                dictionary_separate->
                drillthru_dictionary,
            table_insert_input->
                dictionary_separate->
                prompt_dictionary,
                table_insert_input->
                    dictionary_separate->
                    pair_dictionary,
            table_insert_input->
                dictionary_separate->
                prompt_name_list,
            table_insert_input->security_entity,
            table_insert_input->
                relation_mto1_automatic_preselection,
            query_dictionary->delimited_list );
}
```

TABLE_INSERT_INPUT

VERTICAL_NEW_TABLE

QUERY_DICTIONARY

FORM_TABLE_INSERT_AUTOMATIC

TABLE_INSERT (2)

```
/* Process (continued) */
else
{
   int table_insert_rows_number(
      int table_insert_input->
            folder->
            insert_rows_number,
      VERTICAL_NEW_TABLE *
         table_insert_input->
            vertical_new_table );

   // Create a hidden widget for the many_folder_name
   DICTIONARY *table_insert_prompt_dictionary(
      const char *
         VERTICAL_NEW_PROMPT_MANY_HIDDEN_LABEL,
      DICTIONARY *table_insert_input->
         dictionary_separate->
         prompt_dictionary,
      VERTICAL_NEW_TABLE *
         vertical_new_table );

   FORM_TABLE_INSERT *form_table_insert =
      form_table_insert_new(
         application_name,
         session_key,
         login_name,
         role_name,
         folder_name,
         target_frame,
         table_insert_rows_number(),
         table_insert_input->
            folder->
            post_change_javascript,
         table_insert_input->
            folder_attribute_non_primary_name_list,
         table_insert_input->relation_mto1_list,
         table_insert_input->relation_mto1_isa_list,
         table_insert_input->
            folder_attribute_append_isa_list,
         table_insert_input->
            dictionary_separate->
            drillthru_dictionary,
         table_insert_prompt_dictionary(),
         table_insert_input->
            dictionary_separate->
            query_dictionary,
         table_insert_input->
            dictionary_separate->
            pair_dictionary,
         table_insert_input->
            dictionary_separate->
            ignore_name_list,
         table_insert_input->
            dictionary_separate->
            prompt_name_list,
         table_insert_input->security_entity );
}

char *document_head_calendar_setup_string(
   table_insert_input->
      folder_attribute_calendar_date_name_list_length );

if ( error_string && *error_string )
{
   result_string = (char *)0;
}

char *table_insert_onload_javascript_string(
   VERTICAL_NEW_TABLE *
         vertical_new_table );

LIST *table_insert_javascript_filename_list(
   char *table_insert_input->
      folder->
      javascript_filename );

char *javascript_include_string(
   application_name,
   table_insert_javascript_filename_list() );

DOCUMENT *document =
   document_new(
      application_name,
      application_title_string( application_name ),
      table_insert_input->title_string,
      result_string /* sub_title_string */,
      error_string /* sub_sub_title_string */,
      table_insert_input->folder->notepad,
      table_insert_onload_javascript_string(),
      (table_insert_input->menu)
         ? table_insert_input->menu->html : (char *)0,
      document_head_menu_setup_string(
         (table_insert_input->menu)
            ? 1 : 0 /* menu_boolean */ ),
      document_head_calendar_setup_string(),
      javascript_include_string() );

char *table_insert_form_html(
   FORM_TABLE_INSERT_AUTOMATIC *
      form_table_insert_automatic,
   FORM_TABLE_INSERT *
      form_table_insert );

char *document_form_html(
   document->html,
   document->document_head->html,
   document_head_close_tag(),
   document->document_body->html,
   table_insert_form_html(),
   document_body_close_tag(),
   document_close_tag() );

free( table_insert_form_html() );

/* Attributes */
TABLE_INSERT_INPUT *table_insert_input;
boolean entire_primary_key_ignored;
VERTICAL_NEW_TABLE *vertical_new_table;
QUERY_DICTIONARY *query_dictionary;
FORM_TABLE_INSERT_AUTOMATIC *
   form_table_insert_automatic /* html is freed */;
int rows_number;
DICTIONARY *prompt_dictionary;
FORM_TABLE_INSERT *
   form_table_Insert /* html is freed */;
char *title_string;
char *document_head_calendar_setup_string;
char *onload_javascript_string;
LIST *javascript_filename_list;
char *javascript_include_string;
DOCUMENT *document;
char *document_form_html;
```

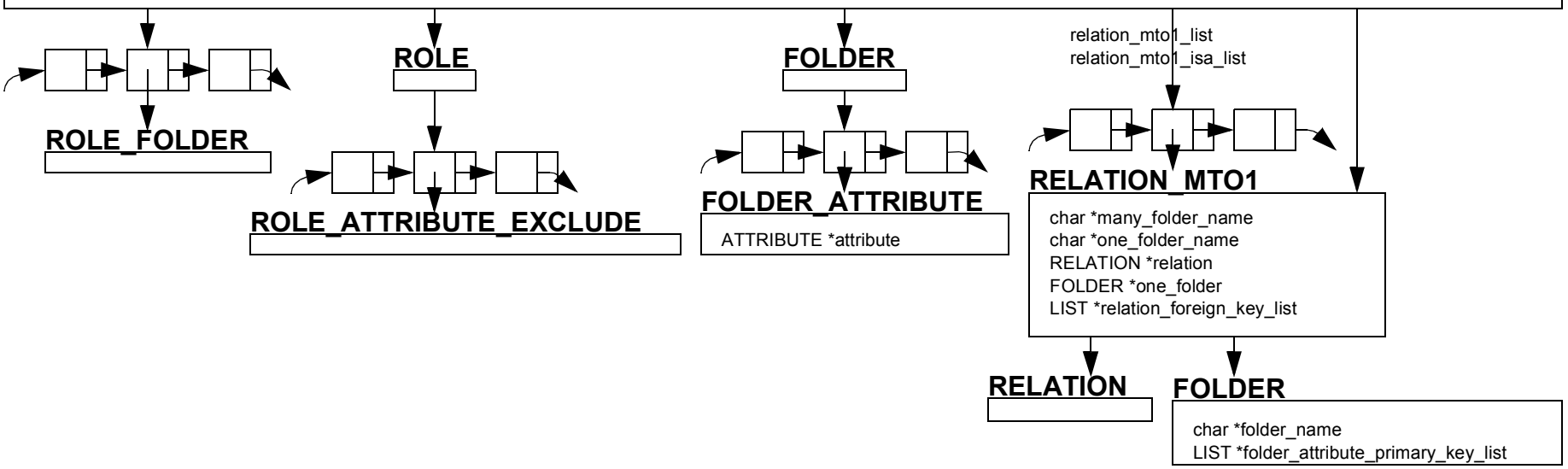
FORM_TABLE_INSERT **DOCUMENT**

TABLE_INSERT (3)

```
/* Usage */
boolean table_insert_forbid(
    boolean role_folder_insert_boolean );

/* Process */
return 1 - role_folder_insert_boolean;

/* Usage */
boolean table_insert_entire_primary_key_ignored(
    LIST *folder_attribute_primary_key_list,
    LIST *ignore_name_list );

/* Process */
LIST *list_subtract(
    folder_attribute_primary_key_list,
    ignore_name_list );

if ( !list_length( list_subtract() ) )
    return 1;
else
    return 0;

/* Driver */
if ( table_insert->
        vertical_new_table )
{
    void vertical_new_table_blank_prompt_frame(
        table_insert->
            vertical_new_table->
            vertical_new_filename->
            output_filename,
        table_insert->
            vertical_new_table->
            document );
}

if ( table_insert->entire_primary_key_ignored )
{
    document_process_output(
        application_name,
        (LIST *)0 /* javascript_filename_list */,
        table_insert->
            table_insert_input->
            title_string );

    printf(
        "<h3>%s</h3>\n",
        TABLE_INSERT_PRIMARY_IGNORED_MESSAGE );

    document_close();
}
else
{
    printf( "%s\n", document_form_html );
}
```

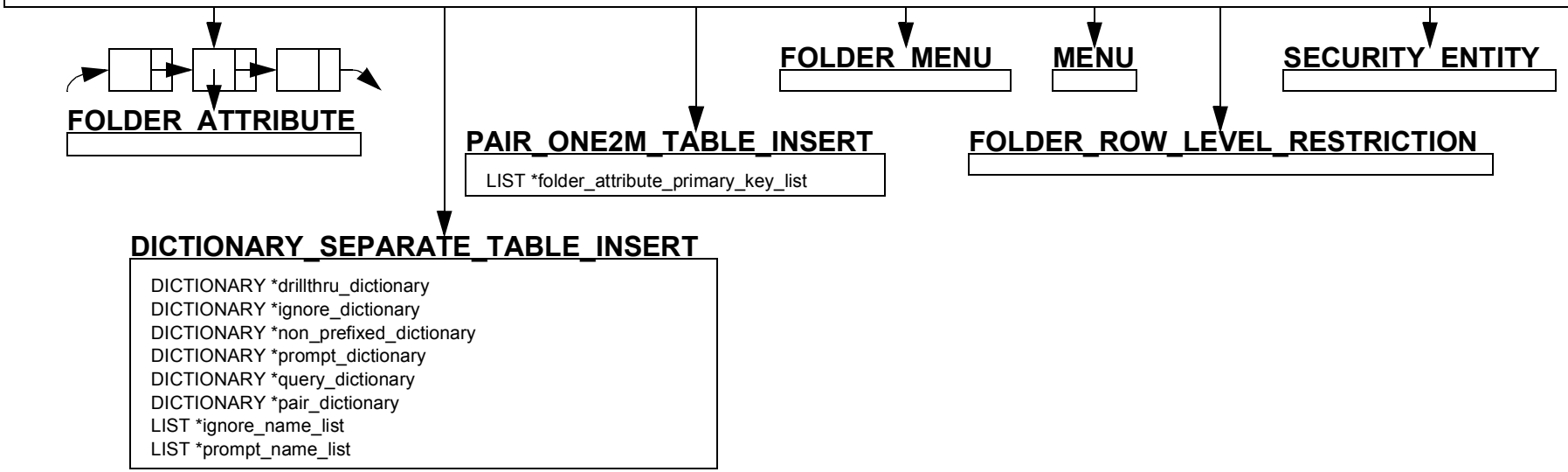
TABLE_INSERT_INPUT (1)

```
/* Usage */
TABLE_INSERT_INPUT *
   table_insert_input_new(
      char *application_name,
      char *session_key,
      char *login_name,
      char *role_name,
      char *folder_name,
      char *target_frame,
      DICTIONARY *original_post_dictionary );

/* Process */
TABLE_INSERT_INPUT *
   table_insert_input_calloc(
      void );

LIST *role_folder_list(
   role_name,
   folder_name );

boolean role_folder_insert_boolean =
   role_folder_state_boolean(
      APPASERVER_INSERT_STATE,
      folder_name,
      role_folder_list() );

boolean table_insert_forbid(
   role_folder_insert_boolean );

if ( table_insert_forbid() ) return this;

char *table_insert_input_title_string(
   char *APPASERVER_INSERT_STATE,
   char *folder_name );

ROLE *role =
   role_fetch(
      role_name,
      1 /* fetch_role_attribute_exclude_list */ );

LIST *role_attribute_exclude_insert_name_list(
   APPASERVER_INSERT_STATE,
   role->role_attribute_exclude_list );

FOLDER *folder =
   folder_fetch(
      folder_name,
      role_attribute_exclude_insert_name_list(),
      /* Sets folder_attribute_primary_list */
      /* Sets folder_attribute_primary_key_list */
      /* Sets folder_attribute_name_list */
      1 /* fetch_folder_attribute_list */,
      1 /* fetch_attribute */,
      1 /* cache_boolean */ );

LIST *folder_attribute_non_primary_name_list(
   folder->folder_attribute_list );

LIST *relation_mto1_list(
   folder_name
      /* many_folder_name */,
   folder->folder_attribute_primary_key_list
      /* many_folder_primary_key_list */ );

RELATION_MTO1 *
   relation_mto1_automatic_preselection(
      relation_mto1_list() );

LIST *relation_mto1_isa_list(
   (LIST *)0 /* mto1_isa_list Pass in null */,
   folder_name
      /* many_folder_name */,
   folder->folder_attribute_primary_key_list
      /* many_folder_primary_key_list */,
   0 /* not fetch_relation_one2m_list */,
   1 /* fetch_relation_mto1_list */ );
```



TABLE_INSERT_INPUT (2)

```
/* Process (continued) */
LIST *folder_attribute_append_isa_list(
    folder->folder_attribute_list /* append_isa_list */,
    relation_mto1_isa_list() );

LIST *folder_attribute_date_name_list(
    folder_attribute_append_isa_list() );

int folder_attribute_date_name_list_length(
    folder_attribute_append_isa_list() );

DICTIONARY_SEPARATE_TABLE_INSERT *
    dictionary_separate =
        dictionary_separate_table_insert_new(
            original_post_dictionary,
            application_name,
            login_name,
            relation_mto1_list(),
            folder_attribute_date_name_list(),
            folder_attribute_append_isa_list(),
            folder->folder_attribute_name_list );

PAIR_ONE2M_TABLE_INSERT *
    pair_one2m_table_insert =
        pair_one2m_table_insert_new(
            PAIR_ONE2M_ONE_FOLDER_KEY,
            dictionary_separate->pair_dictionary );

char *appaserver_parameter_data_directory();

boolean application_menu_horizontal_boolean(
    application_name );

boolean vertical_new_table_participating_boolean(
    VERTICAL_NEW_PROMPT_ONE_HIDDEN_LABEL,
    dictionary_separate->non_prefixed_dictionary );

boolean table_insert_input_menu_horizontal_boolean(
    FRAMESET_PROMPT_FRAME,
    target_frame,
    application_menu_horizontal_boolean(),
    vertical_new_table_participating_boolean() );

if (    !vertical_new_table_participating_boolean()
&&  table_insert_input_menu_horizontal_boolean() )
{
    FOLDER_MENU *folder_menu =
        folder_menu_fetch(
            application_name,
            session_key,
            role_name,
            appaserver_parameter_data_directory(),
            0 /* not folder_menu_remove_boolean */ );

    MENU *menu =
        menu_new(
            application_name,
            session_key,
            login_name,
            role_name,
            1 /* application_menu_horizontal_boolean */,
            folder_menu->count_list );
}

FOLDER_ROW_LEVEL_RESTRICTION *
    folder_row_level_restriction =
        folder_row_level_restriction_fetch(
            folder_name );

SECURITY_ENTITY *security_entity =
    security_entity_new(
        login_name,
        folder_name,
        folder_row_level_restriction->
            non_owner_forbid,
        role->override_row_restrictions );
```

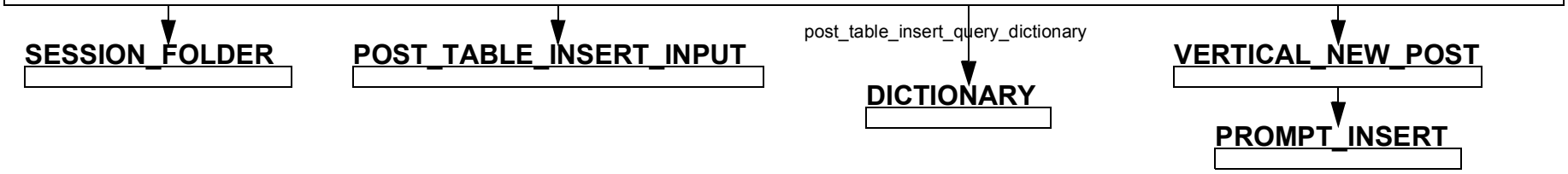


TABLE_INSERT_INPUT (3)

```
/* Usage */
boolean table_insert_input_menu_horizontal_boolean(
    const char *FRAMESET_PROMPT_FRAME,
    char *target_frame,
    boolean application_menu_horizontal_boolean(),
    boolean vertical_new_table_participating_boolean );

/* Process */
if ( vertical_new_table_participating_boolean )
{
    menu_horizontal_boolean =
        application_menu_horizontal_boolean;
}
else
{
    boolean menu_horizontal_boolean(
        target_frame,
        FRAMESET_PROMPT_FRAME,
        application_menu_horizontal_boolean );
}

/* Attributes */
LIST *role_folder_list;
boolean role_folder_insert_boolean;
boolean table_insert_forbid;
char *title_string;
ROLE *role;
LIST *role_attribute_exclude_insert_name_list;
FOLDER *folder;
LIST *folder_attribute_non_primary_name_list;
LIST *relation_mto1_list;
RELATION_MTO1 *
    relation_mto1_automatic_preselection;
LIST *relation_mto1_isa_list;
LIST *folder_attribute_append_isa_list;
LIST *folder_attribute_date_name_list;
int folder_attribute_calendar_date_name_list_length;
DICTIONARY_SEPARATE_TABLE_INSERT *
    dictionary_separate;
PAIR_ONE2M_TABLE_INSERT *pair_one2m_table_insert;
char *appaserver_parameter_data_directory;
boolean application_menu_horizontal_boolean;
boolean vertical_new_table_participating_boolean;
boolean menu_horizontal_boolean;
FOLDER_MENU *folder_menu;
MENU *menu;
FOLDER_ROW_LEVEL_RESTRICTION *
    folder_row_level_restriction;
SECURITY_ENTITY *security_entity;
```

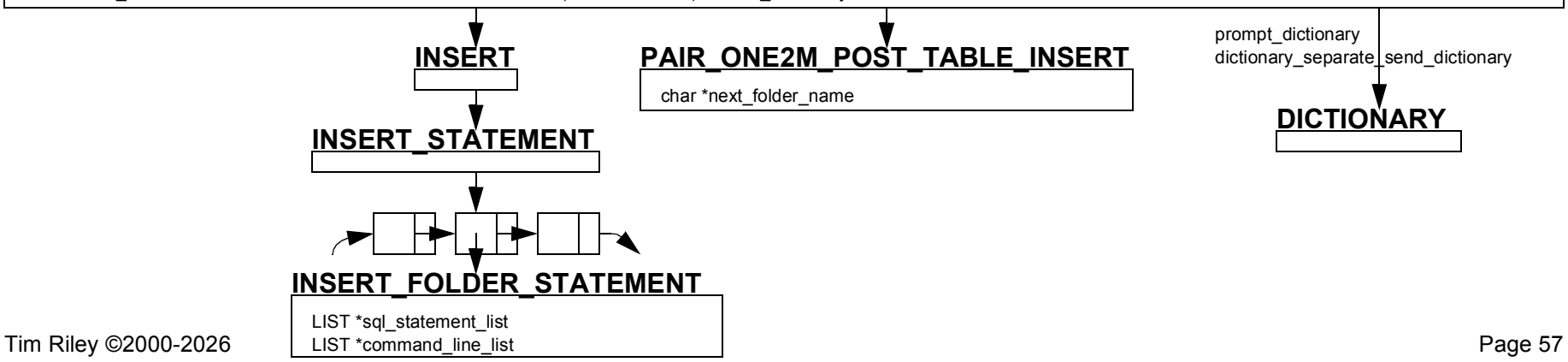
POST_TABLE_INSERT (1)

```
/* Usage */
POST_TABLE_INSERT *post_table_insert_new(
    int argc,
    char **argv,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name );

/* Process */
POST_TABLE_INSERT *post_table_insert_calloc(
    void );

SESSION_FOLDER *session_folder =
    /* Sets appaserver environment and outputs argv */
    /* Each parameter is security inspected */
    session_folder_integrity_exit(
        argc,
        argv,
        application_name,
        session_key,
        login_name,
        role_name,
        folder_name,
        APPASERVER_INSERT_STATE,
        (char *)0 /* state2 */ );

POST_TABLE_INSERT_INPUT *
    post_table_insert_input =
        post_table_insert_input_new(
            application_name,
            session_key,
            login_name,
            role_name,
            folder_name );

if ( post_table_insert_input->insert_appaserver_user )
{
    dictionary_set(
        post_table_insert_input->
            dictionary_separate->
            prompt_dictionary,
        APPASERVER_USER_FULL_NAME_COLUMN,
        post_table_insert_input->
            insert_appaserver_user->
            full_name );

    dictionary_set(
        post_table_insert_input->
            dictionary_separate->
            prompt_dictionary,
        APPASERVER_USER_CONTACT_KEY_COLUMN,
        post_table_insert_input->
            insert_appaserver_user->
            contact_key );
}

VERTICAL_NEW_POST *vertical_new_post =
    vertical_new_post_new(
        VERTICAL_NEW_PROMPT_MANY_HIDDEN_LABEL,
        FRAMESET_PROMPT_FRAME,
        DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
        application_name,
        session_key,
        login_name,
        role_name,
        post_table_insert_input->
            dictionary_separate->
            drillthru_dictionary,
        post_table_insert_input->
            dictionary_separate->
            non_prefixed_dictionary,
        post_table_insert_input->
            appaserver_parameter->
            data_directory );

DICTIONARY *post_table_insert_query_dictionary(
    QUERY_RELATION_OPERATOR_PREFIX,
    QUERY_EQUAL,
    post_table_insert_input->
        dictionary_separate->
        prompt_dictionary,
    post_table_insert_input->
        dictionary_separate->
        multi_row_dictionary,
    vertical_new_post );

DICTIONARY *post_table_insert_prompt_dictionary(
    const char *TABLE_EDIT_ONLOAD_JAVASCRIPT_KEY,
    DICTIONARY *
        post_table_insert_input->
            dictionary_separate->
            prompt_dictionary /* in/out */,
    VERTICAL_NEW_POST *vertical_new_post );
```



POST_TABLE_INSERT (2)

```
/* Process (continued) */
if ( post_table_insert->
       post_table_insert_input->
          relation_mto1_automatic_preselection )
{
   INSERT *insert =
      insert_automatic_new(
         application_name,
         session_key,
         login_name,
         role_name,
         folder_name,
         post_table_insert_input->
            folder->
            folder_attribute_primary_key_list,
         post_table_insert_input->
            folder_attribute_append_isa_list,
         post_table_insert_input->
            relation_mto1_isa_list,
         post_table_insert_input->
            dictionary_separate->
            prompt_dictionary,
         post_table_insert_input->
            dictionary_separate->
            multi_row_dictionary,
         post_table_insert_input->
            post_change_process );
}
else
{
   INSERT *insert =
      insert_new(
         application_name,
         session_key,
         login_name,
         role_name,
         folder_name,
         post_table_insert_input->
            folder->
            folder_attribute_primary_key_list
            /* root_primary_key_list */,
         post_table_insert_input->
            folder_attribute_append_isa_list,
         post_table_insert_input->
            relation_mto1_list,
         post_table_insert_input->
            relation_mto1_isa_list,
         post_table_insert_input->
            dictionary_separate->
            prompt_dictionary
            /* in/out */,
         post_table_insert_input->
            dictionary_separate->
            multi_row_dictionary
            /* in/out */,
         post_table_insert_input->
            dictionary_separate->
            ignore_name_list,
         post_table_insert_input->
            post_change_process );
}

PAIR_ONE2M_POST_TABLE_INSERT *
   pair_one2m_post_table_insert =
      pair_one2m_post_table_insert_new(
         post_table_insert_input->
            dictionary_separate->
            pair_dictionary,
         folder_name /* many_folder_name */ );

DICTIONARY *dictionary_separate_send_dictionary(
   (DICTIONARY *)0 /* sort_dictionary */,
   DICTIONARY_SEPARATE_SORT_PREFIX,
   post_table_insert_query_dictionary(),
   DICTIONARY_SEPARATE_QUERY_PREFIX,
   post_table_insert_input->
      dictionary_separate->
      drillthru_dictionary,
   DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
   post_table_insert_input->
      dictionary_separate->
      ignore_dictionary,
   DICTIONARY_SEPARATE_IGNORE_PREFIX,
   (DICTIONARY *)0 /* no_display_dictionary */,
   DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
   post_table_insert_input->
      dictionary_separate->
      pair_dictionary,
   DICTIONARY_SEPARATE_PAIR_PREFIX,
   post_table_insert_prompt_dictionary()
      /* non_prefixed_dictionary */ );

char *dictionary_separate_send_string(
   dictionary_separate_send_dictionary() );

/* Attributes */
SESSION_FOLDER *session_folder;
POST_TABLE_INSERT_INPUT *post_table_insert_input;
VERTICAL_NEW_POST *vertical_new_post;
DICTIONARY *query_dictionary;
DICTIONARY *prompt_dictionary;
INSERT *insert;
PAIR_ONE2M_POST_TABLE_INSERT *
   pair_one2m_post_table_insert;
DICTIONARY *dictionary_separate_send_dictionary;
char *dictionary_separate_send_string;
```



POST_TABLE_INSERT (3)

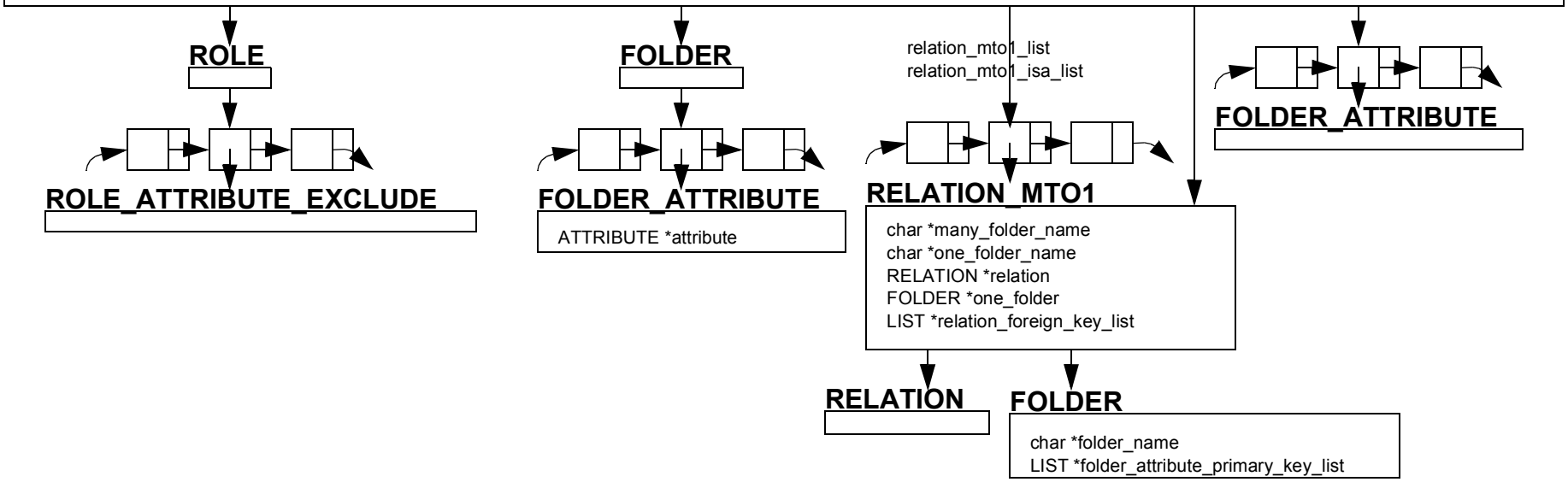
```
/* Usage */
DICTIONARY *post_table_insert_query_dictionary(
    const char *QUERY_RELATION_OPERATOR_PREFIX,
    const char *QUERY_EQUAL,
    DICTIONARY *prompt_dictionary,
    DICTIONARY *multi_row_dictionary,
    VERTICAL_NEW_POST *vertical_new_post );

/* Process */
if ( !vertical_new_post )
{
    return
    DICTIONARY *post_prompt_insert_query_dictionary(
        QUERY_RELATION_OPERATOR_PREFIX,
        QUERY_EQUAL,
        prompt_dictionary );
}
DICTIONARY *dictionary_remove_index(
    multi_row_dictionary );

DICTIONARY *post_prompt_insert_query_dictionary(
    QUERY_RELATION_OPERATOR_PREFIX,
    QUERY_EQUAL,
    dictionary_remove_index() );
```

POST_TABLE_INSERT (4)

```
/* Driver */
if ( post_table_insert->insert )
{
    char *insert_statement_error_string =
        char *insert_statement_execute(
            application_name,
            post_table_insert->
                insert->
                insert_statement->
                insert_folder_statement_list,
            post_table_insert->
                insert->
                appaserver_error_filespecification );

    char *result_string =
        post_table_insert->
            insert->
            results_string;
}
else
{
    char *result_string = "No Rows Inserted";
}

if ( post_table_insert->vertical_new_post )
{
    post_table_insert->
        vertical_new_post->
        prompt_insert =
            prompt_insert_new(
                application_name,
                session_key,
                login_name,
                role_name,
                post_table_insert->
                    vertical_new_post->
                    many_folder_name,
                post_table_insert->
                    vertical_new_post->
                    application_menu_horizontal_boolean,
                post_table_insert->
                    vertical_new_post->
                    data_directory,
                post_table_insert->
                    vertical_new_post->
                    original_post_dictionary );

    vertical_new_post_prompt_insert_create(
        post_table_insert->
            vertical_new_post->
            vertical_new_filename,
        post_table_insert->
            vertical_new_post->
            prompt_insert );
}

if (    post_table_insert->
            pair_one2m_post_table_insert
&&  post_table_insert->
            pair_one2m_post_table_insert->
            next_folder_name )
{
    char *system_string =
        char *execute_system_string_table_insert(
            TABLE_INSERT_EXECUTABLE,
            session_key,
            login_name,
            role_name,
            post_table_insert->
                pair_one2m_post_table_insert->
                next_folder_name,
            target_frame,
            results_string,
            insert_statement_error_string,
            post_table_insert->
                dictionary_separate_send_string,
            appaserver_error_filespecification(
                application_name ) );

    goto execute_system_string;
}
else
if (    post_table_insert->
            pair_one2m_post_table_insert
&&  !post_table_insert->
            pair_one2m_post_table_insert->
            next_folder_name )
{
    char *one_folder_name =
        dictionary_get(
            PAIR_ONE2M_ONE_FOLDER_KEY,
            post_table_insert->
                post_table_insert_input->
                dictionary_separate->
                    pair_dictionary );

    snprintf(
        pair_one2m_result_string[],
        sizeof ( pair_one2m_results_string ),
        "%s<br>Insert Complete",
        (result_string) ? result_string : "" );

    char *system_string =
        char *execute_system_string_table_edit(
            TABLE_EDIT_EXECUTABLE,
            session_key,
            login_name,
            role_name,
            one_folder_name,
            target_frame,
            pair_one2m_result_string,
            insert_statement_error_string,
            post_table_insert->
                dictionary_separate_send_string,
            appaserver_error_filespecification(
                application_name );

    goto execute_system_string;
}
else
{
    char *system_string =
        char *execute_system_string_table_edit(
            TABLE_EDIT_EXECUTABLE,
            session_key,
            login_name,
            role_name,
            folder_name,
            target_frame,
            result_string,
            insert_statement_error_string,
            post_table_insert->
                dictionary_separate_send_string,
            appaserver_error_filespecification(
                application_name );
}

execute_system_string:
system( system_string );
```

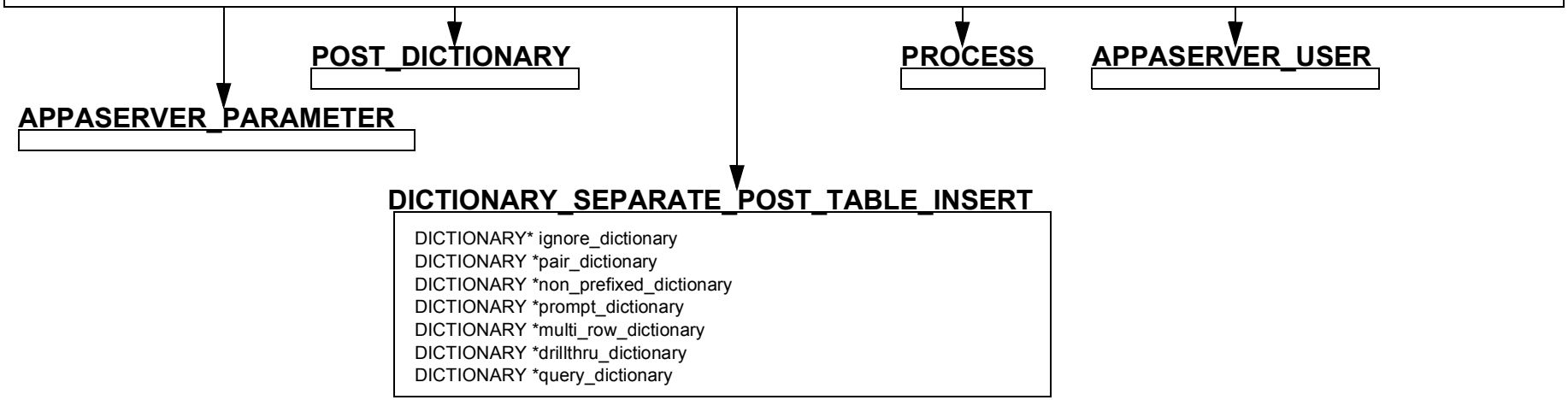
POST_TABLE_INSERT_INPUT (1)

```
/* Usage */
POST_TABLE_INSERT_INPUT *
    post_table_insert_input_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name );

/* Process */
POST_TABLE_INSERT_INPUT *
    post_table_insert_input_calloc(
        void );

ROLE *role =
    role_fetch(
        role_name,
        1 /* fetch_role_attribute_exclude_list */ );

LIST *role_attribute_exclude_insert_name_list(
    APPASERVER_INSERT_STATE,
    role->role_attribute_exclude_list );

FOLDER *folder =
    folder_fetch(
        folder_name,
        role_attribute_exclude_insert_name_list(),
        /* Sets folder_attribute_primary_list */
        /* Sets folder_attribute_primary_key_list */
        /* Sets folder_attribute_name_list */
        1 /* fetch_folder_attribute_list */,
        1 /* fetch_attribute */,
        1 /* cache_boolean */ );

LIST *relation_mto1_list(
    folder_name
        /* many_folder_name */,
    folder->folder_attribute_primary_key_list
        /* many_folder_primary_key_list */ );

RELATION_MTO1 *
    relation_mto1_automatic_preselection(
        relation_mto1_list() );

LIST *relation_mto1_isa_list(
    (LIST *)0 /* mto1_isa_list Pass in null */,
    folder_name
        /* many_folder_name */,
    folder->folder_attribute_primary_key_list
        /* many_primary_key_list */,
    0 /* not fetch_relation_one2m_list */,
    0 /* not fetch_relation_mto1_list */ );

LIST *folder_attribute_append_isa_list(
    folder->folder_attribute_list,
    relation_mto1_isa_list() );

LIST *folder_attribute_name_list(
    (char *)0 /* folder_name */,
    folder_attribute_append_isa_list() );

LIST *folder_attribute_date_name_list(
    folder_attribute_append_isa_list() );

LIST *folder_attribute_upload_filename_list(
    folder_attribute_append_isa_list() );
```



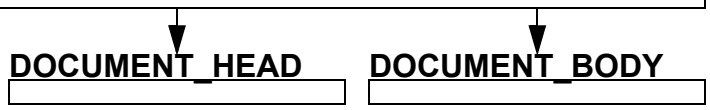
POST_TABLE_INSERT_INPUT (2)

```
/* Process (continued) */
APPASERVER_PARAMETER *
    appaserver_parameter =
        appaserver_parameter_new();

POST_DICTIONARY *post_dictionary =
    post_dictionary_stdin_new(
        /* Used when expecting a spooled file */
        WIDGET_UPLOAD_RECALL_PREFIX,
        application_name,
        appaserver_parameter->upload_directory,
        folder_attribute_upload_filename_list() );

DICTIONARY_SEPARATE_POST_TABLE_INSERT *
    dictionary_separate =
        dictionary_separate_post_table_insert_new(
            post_dictionary->original_post_dictionary,
            application_name,
            login_name,
            relation_mto1_list(),
            folder_attribute_name_list(),
            folder_attribute_date_name_list(),
            folder_attribute_append_isa_list() );

if ( folder->post_change_process_name )
{
    PROCESS *post_change_process =
        process_fetch(
            folder->post_change_process_name,
            (char *)0 /* document_root */,
            (char *)0 /* relative_source */,
            1 /* check_executable_inside_filesystem */,
            appaserver_parameter->mount_point );
}

APPASERVER_USER *insert_appaserver_user(
    APPASERVER_USER_FULL_NAME_COLUMN,
    login_name,
    role_attribute_exclude_insert_name_list(),
    folder_attribute_append_isa_list() );

/* Attributes */
ROLE *role;
LIST *role_attribute_exclude_insert_name_list;
FOLDER *folder;
LIST *relation_mto1_list;
RELATION_MTO1 *
    relation_mto1_automatic_preselection;
LIST *relation_mto1_isa_list;
LIST *folder_attribute_append_isa_list;
LIST *folder_attribute_name_list;
LIST *folder_attribute_date_name_list;
LIST *folder_attribute_upload_filename_list;
APPASERVER_PARAMETER *appaserver_parameter;
POST_DICTIONARY *post_dictionary;
DICTIONARY_SEPARATE_POST_TABLE_INSERT *
    dictionary_separate;
PROCESS *post_change_process;
APPASERVER_USER *insert_appaserver_user;
```



DOCUMENT (1)

```
/* Usage */
DOCUMENT *document_new(
    char *application_name,
    char *application_title_string(),
    char *title_string,
    char *sub_title_string,
    char *sub_sub_title_string,
    char *notepad,
    char *onload_javascript_string,
    char *menu_html,
    char *document_head_menu_setup_string(
        menu_boolean ),
    char *document_head_calendar_setup_string(
        folder_attribute_date_name_list_length ),
    char *javascript_include_string() );

/* Process */
DOCUMENT *document_calloc(
    void );

DOCUMENT_HEAD *document_head =
    document_head_new(
        application_name,
        application_title_string,
        document_head_menu_setup_string,
        document_head_calendar_setup_string,
        javascript_include_string );

DOCUMENT_BODY *document_body =
    document_body_new(
        menu_html,
        title_string,
        sub_title_string,
        sub_sub_title_string,
        notepad,
        onload_javascript_string );

char *document_type_html(
    void );

char *document_standard_string(
    void );

char *document_html(
    char *document_type_html(),
    char *document_standard_string() );

/* Usage */
DOCUMENT *document_basic_new(
    char *application_name,
    LIST *javascript_filename_list,
    char *application_title_string() );

/* Process */
DOCUMENT *document_calloc();

char *document_type_html();

char *document_standard_string();

char *document_html(
    document_type_html(),
    document_standard_string() );

char *javascript_include_string(
    application_name,
    javascript_filename_list );

DOCUMENT_HEAD *document_head =
    document_head_new(
        application_name,
        application_title_string,
        (char *)0 /* menu_setup_string */,
        (char *)0 /* calendar_setup_string */,
        javascript_include_string() );

DOCUMENT_BODY *document_body =
    document_body_new(
        (char *)0 /* menu_html */,
        (char *)0 /* title_string */,
        (char *)0 /* sub_title_string */,
        (char *)0 /* sub_sub_title_string */,
        (char *)0 /* notepad */,
        (char *)0 /* onload_javascript_string */ );

/* Attributes */
char *type_html;
char *standard_string;
char *javascript_include_string;
DOCUMENT_HEAD *document_head;
LIST *widget_notepad_container_list;
DOCUMENT_BODY *document_body;
char *html;
```



DOCUMENT (2)

```
/* Usage */
void document_process_output(
    char *application_name,
    LIST *javascript_filename_list,
    char *title_string );

/* Process */
DOCUMENT *document =
    document_basic_new(
        application_name,
        javascript_filename_list,
        application_title_string(
            application_name ) );

void document_content_type_output();

printf("%s\n%s\n%s\n%s\n",
       document->html,
       document->document_head->html,
       document_head_close_tag(),
       document->document_body->html );

char *document_process_title_string(
    title_string );

if ( title_string )
{
    char *document_process_sub_title_string();

    printf("<h1>%s</h1>\n<h2>%s</h2>\n",
           document_process_title_string(),
           document_process_sub_title_string() );
}

/* Usage */
char *document_process_title_string(
    char *title_string );

/* Process */
if ( !process_name ) return (char *)0;

char *string_initial_capital(
    static string[],
    title_string );

/* Usage */
char *document_process_sub_title_string(
    void );

/* Process */
char *date_now_yyyy_mm_dd_hhmm(
    date_utc_offset() );

/* Usage */
char *document_form_html(
    char *document->html,
    char *document->document_head->html,
    char *document_head_close_tag(),
    char *document->document_body->html,
    char *form_html,
    char *document_body_close_tag(),
    char *document_close_tag() );

/* Process */

/* Usage */
char *document_content_type_html( void );

/* Process */
return "Content-type text/html\n";

/* Usage */
void document_content_type_output( void );

/* Process */
static boolean did_already;

if ( did_already ) return;

printf(
    "%s\n",
    document_content_type_html() );

fflush( stdout );

did_already = 1;

/* Usage */
char *document_script_close_tag( void );

/* Process */
return "</script>";

/* Usage */
char *document_close_tag( void );

/* Process */
return "</html>";

/* Usage */
char *document_both_close_tag( void );

/* Process */
snprintf(
    static both_close_tag[],
    sizeof ( both_close_tag ),
    "%s\n%s",
    document_body_close_tag(),
    document_close_tag() );

/* Usage */
void document_close( void );

/* Process */
printf(
    "%s\n",
    document_both_close_tag() );

/* Usage */
char *document_capacity_message(
    const char *function_name,
    const int max_strlen,
    const double APPASERVER_CAPACITY_THRESHOLD,
    char *label,
    int strlen_html );

/* Process */
double document_capacity(
    const int max_strlen,
    int strlen_html );

if (    document_capacity() <
        APPASERVER_CAPACITY_THRESHOLD )
{
    return NULL;
}
```

DOCUMENT_HEAD

```
/* Usage */
DOCUMENT_HEAD *document_head_new(
    char *application_name,
    char *application_title_string,
    char *document_head_menu_setup_string(),
    char *document_head_calendar_setup_string(),
    char *javascript_include_string() );

/* Process */
DOCUMENT_HEAD *document_head_calloc(
    void );

char *document_head_meta_string(
    void );

char *document_head_stylesheet_string(
    void );

char *document_head_title_tag(
    char *application_title_string );

char *document_head_html(
    document_head_meta_string(),
    document_head_stylesheet_string(),
    document_head_title_tag(),
    document_head_menu_setup_string,
    document_head_calendar_setup_string,
    javascript_include_string );
```

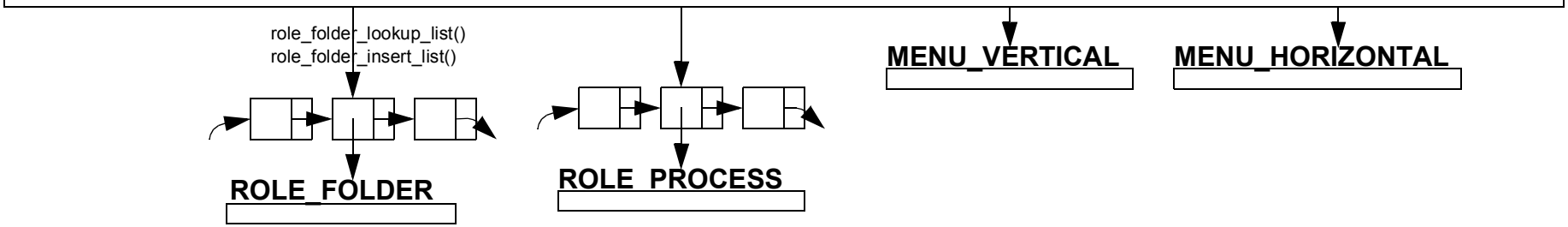
```
/* Usage */
char *document_head_html(
    char *document_head_meta_string,
    char *document_head_stylesheet_string,
    char *document_head_title_tag,
    char *document_head_menu_setup_string(),
    char *document_head_calendar_setup_string(),
    char *javascript_include_string );

/* Process */
char *document_head_open_tag( void );

/* Usage */
void document_head_output_html_stream(
    FILE *output_stream );

/* Process */
char *environment_application_name();

DOCUMENT *document =
    document_basic_new(
        environment_application_name(),
        (LIST *)0 /* javascript_filename_list */,
        application_title_string(
            environment_application_name() ) );

fprintf(output_stream,
        "%s\n%s\n",
        document->html,
        document->document_head->html );

/* Public */
char *document_head_menu_setup_string(
    boolean menu_boolean );

char *document_head_calendar_setup_string(
    folder_attribute_date_name_list_length() );

char *document_head_close_tag( void );

void document_head_output_stylesheet(
    FILE *output_stream,
    char *application_name,
    char *stylesheet_filename );

char *document_head_javascript_string(
    void );

/* Attributes */
char *meta_string;
char *stylesheet_string;
char *title_tag;
char *html;
```

DOCUMENT_BODY

```
/* Usage */
DOCUMENT_BODY *document_body_new(
    char *menu_html,
    char *title_string,
    char *sub_title_string,
    char *sub_sub_title_string,
    char *notepad,
    char *onload_javascript_string );

/* Process */
char *document_body_onload_open_tag(
    JAVASCRIPT_WAIT_OVER,
    onload_javascript_string,
    document_body_menu_onload_string() );

if ( menu_html )
{
    char *document_body_menu_onload_string(
        void );

    char *document_body_hide_preload_html(
        void );
}

if ( notepad )
{
    char *document_body_notepad_html(
        notepad );
}

char *document_body_html(
    char *menu_html,
    char *title_string,
    char *sub_title_string,
    char *sub_sub_title_string,
    char *document_body_onload_open_tag()
        /* body_onload_open_tag */,
    char *document_body_hide_preload_html(),
    char *document_body_notepad_html() );

/* Usage */
char *document_body_notepad_html(
    char *notepad );

/* Process */
char *style =
"{text-align: left; padding: 0 3em 0 3em}";

sprintf(
    notepad_html[],
    "<p style=\"%s\">%s",
    style,
    notepad );

/* Usage */
char *document_body_onload_open_tag(
    const char *JAVASCRIPT_WAIT_OVER,
    char *onload_javascript_string,
    char *document_body_menu_onload_string() );

/* Process */
char tag[];
char *ptr = tag;

ptr += sprintf( ptr,
    "<body leftmargin=0 "
    "topmargin=0 "
    "marginwidth=0 "
    "marginheight=0" );

    ptr += sprintf( ptr,
        " onload=\"%s",
        JAVASCRIPT_WAIT_OVER );

if ( menu_onload_string )
{
    ptr += sprintf( ptr,
        " && %s",
        menu_onload_string );
}

if ( onload_javascript_string )
{
    ptr += sprintf( ptr,
        " && %s",
        onload_javascript_string );
}

ptr += sprintf( ptr, "\">" );

/* Usage */
char *document_body_set_onload_javascript_string(
    char *javascript_string,
    char *append_javascript_string );

/* Process */

/* Usage */
char *document_body_open_tag( void );

/* Process */
return "<body>";

/* Usage */
char *document_body_close_tag( void );

/* Process */
return "</body>";

/* Attributes */
char *menu_onload_string;
char *onload_open_tag;
char *hide_preload_html;
char *notepad_html;
char *html;
```

MENU

```
/* Usage */
MENU *menu_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    boolean application_menu_horizontal_boolean(),
    LIST *folder_menu_count_list );

/* Process */
MENU *menu_calloc(
    void );

LIST *role_folder_lookup_list( role_name );

LIST *role_folder_insert_list( role_name );

LIST *role_process_list(
    role_name,
    1 /* fetch_process */ );

LIST *appaserver_user_role_name_list(
        login_name );

LIST *role_process_group_name_list(
    role_process_list(),

LIST *role_folder_subschema_name_list(
    role_folder_lookup_list() );

LIST *role_folder_subschema_missing_folder_name_list(
    role_folder_lookup_list() );

LIST *role_process_missing_group_name_list(
    role_process_list() );

if ( !application_menu_horizontal_boolean )
{
    MENU_VERTICAL *
        menu_vertical_new(
            application_name,
            session_key,
            login_name,
            role_name,
            folder_menu_count_list(),
            role_folder_lookup_list(),
            role_folder_insert_list(),
            role_process_list(),
            appaserver_user_role_name_list(),
            role_folder_subschema_name_list(),
            role_folder_subschema_missing_folder_name_list(),
            role_process_group_name_list(),
            role_process_set_missing_group_name_list() );
}
else
{
    MENU_HORIZONTAL *
        menu_horizontal_new(
            application_name,
            session_key,
            login_name,
            role_name,
            folder_menu_count_list(),
            role_folder_lookup_list(),
            role_folder_insert_list(),
            role_process_list(),
            appaserver_user_role_name_list(),
            role_folder_subschema_name_list(),
            role_folder_subschema_missing_folder_name_list(),
            role_process_group_name_list(),
            role_process_missing_group_name_list() );
}

char *menu_html(
    MENU_VERTICAL *menu_vertical_new(),
    MENU_HORIZONTAL *menu_horizontal_new() );

/* Usage */
boolean menu_horizontal_boolean(
    char *current_frame,
    char *FRAMESET_PROMPT_FRAME,
    boolean application_menu_horizontal_boolean() );

/* Process */
return
(   application_menu_horizontal_boolean &&
    string_strcmp(
        current_frame,
        FRAMESET_PROMPT_FRAME ) == 0 );

/* Driver */
printf( "%s\n", menu->html );

/* Attributes */
LIST *role_folder_lookup_list;
LIST *role_folder_insert_list;
LIST *role_process_list;
LIST *appaserver_user_role_name_list;
LIST *role_folder_subschema_name_list;
LIST *role_folder_subschema_missing_folder_name_list;
LIST *role_process_group_name_list;
LIST *role_process_missing_group_name_list;
MENU_VERTICAL *menu_vertical;
MENU_HORIZONTAL *menu_horizontal;
char *html;
```

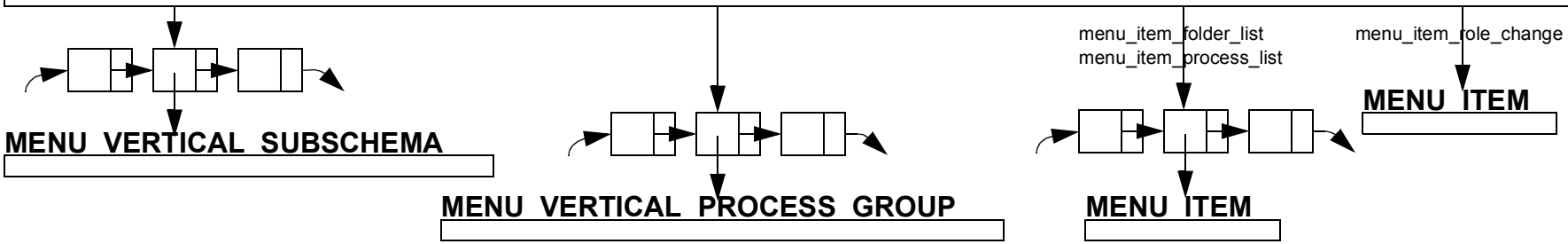


MENU_VERTICAL (1)

```
/* Usage */
MENU_VERTICAL *
   menu_vertical_new(
      char *application_name,
      char *session_key,
      char *login_name,
      char *role_name,
      LIST *folder_menu_count_list,
      LIST *role_folder_lookup_list,
      LIST *role_folder_insert_list,
      LIST *role_process_list,
      LIST *appaserver_user_role_name_list,
      LIST *role_folder_subschema_name_list,
      LIST *role_folder_subschema_missing_folder_name_list,
      LIST *role_process_group_name_list,
      LIST *role_process_missing_group_name_list );

/* Process */
LIST *menu_vertical_subschema_list = list_new();

for char *subschema_name in
   role_folder_subschema_name_list
{
   list_set(
      menu_vertical_subschema_list,
      menu_vertical_subschema_new(
         application_name,
         session_key,
         login_name,
         role_name,
         folder_menu_count_list(),
         role_folder_lookup_list(),
         role_folder_insert_list(),
         subschema_name ) );
}

LIST *menu_item_folder_list = list_new();

for char *folder_name in
   role_folder_subschema_missing_folder_name_list
{
   list_set(
      menu_item_folder_list,
      menu_item_vertical_folder_new(
         application_name,
         session_key,
         login_name,
         role_name,
         folder_name,
         (folder_menu_count_list)
            ? folder_menu_count_seek(
               folder_name,
               folder_menu_count_list)->display
            : (char *)0,
         role_folder_insert_exists(
            folder_name,
            role_folder_insert_list ) );
}

LIST *menu_vertical_process_group_list = list_new();

for char *group_name in
   role_process_group_name_list
{
   list_set(
      menu_vertical_process_group_list,
      menu_vertical_process_group_new(
         application_name,
         session_key,
         login_name,
         role_name ,
         group_name,
         role_process_list );
}

LIST *menu_item_process_list = list_new();

for char *process_name in
   role_process_missing_group_name_list
{
   list_set(
      menu_item_process_list,
      menu_item_vertical_process_new(
         application_name,
         session_key,
         login_name,
         role_name,
         process_name ) );
}

if ( list_length( appaserver_user_role_name_list ) > 1 )
{
   MENU_ITEM *menu_item_role_change =
      menu_item_vertical_role_change_new(
         application_name,
         session_key,
         login_name,
         menu_item_role_name_list(
               role_name,
            appaserver_user_role_name_list() );
}

char *menu_vertical_html(
   login_name,
   role_name,
   menu_vertical_subschema_list,
   menu_item_folder_list,
   menu_vertical_process_group_list,
   menu_item_process_list,
   (menu_item_role_change)
      ? menu_item_role_change->html
      : (char *)0 );
```

MENU_VERTICAL (2)

```
/* Usage */
/* Note: frees each ->html */
char *menu_vertical_html(
    char *login_name,
    char *role_name,
    LIST *menu_vertical_subschema_list,
    LIST *menu_item_folder_list,
    LIST *menu_vertical_process_group_list,
    LIST *menu_item_process_list,
    char *menu_item_role_change_html );

/* Process */
char html[];
char *ptr = html;

char *menu_vertical_html_title(
    char *login_name,
    char *role_name );

ptr += sprintf( ptr,
    "%s",
    menu_vertical_html_title() );

ptr += sprintf( ptr, "<table border=1>\n" );

for MENU_VERTICAL_SUBSCHEMA *
    menu_vertical_subschema in
        menu_vertical_subschema_list
{
    ptr += sprintf( ptr,
        "%s\n",
        menu_vertical_subschema->html );

    free( menu_vertical_subschema->html );
}

if ( list_rewind( menu_item_folder_list ) )
{
    ptr += sprintf( ptr,
        "<tr><th colspan=3>Unassigned\n" );
}

for MENU_ITEM *menu_item in menu_item_folder_list
{
    ptr += sprintf( ptr,
        "%s\n",
        menu_item->html );

    free( menu_item->html );
}

for MENU_VERTICAL_PROCESS_GROUP *
    menu_vertical_process_group in
        menu_vertical_process_group_list
{
    ptr += sprintf( ptr,
        "%s\n",
        menu_vertical_process_group->html );

    free( menu_vertical_process_group->html );
}

for MENU_ITEM *menu_item in menu_item_process_list
{
    ptr += sprintf( ptr,
        "%s\n",
        menu_item->html );

    free( menu_item->html );
}

ptr += sprintf( ptr, "</table>\n" );

if ( menu_item_role_change_html )
{
    ptr += sprintf( ptr,
        "<h3>Role Change</h3>\n" );

    sprintf( ptr, "%s\n", menu_item_role_change_html );

    free( menu_item_role_change_html );
}

/* Attributes */
LIST *menu_vertical_subschema_list;
LIST *menu_item_folder_list;
LIST *menu_vertical_process_group_list;
LIST *menu_item_process_list;
MENU_ITEM *menu_item_role_change;
char *html;
```

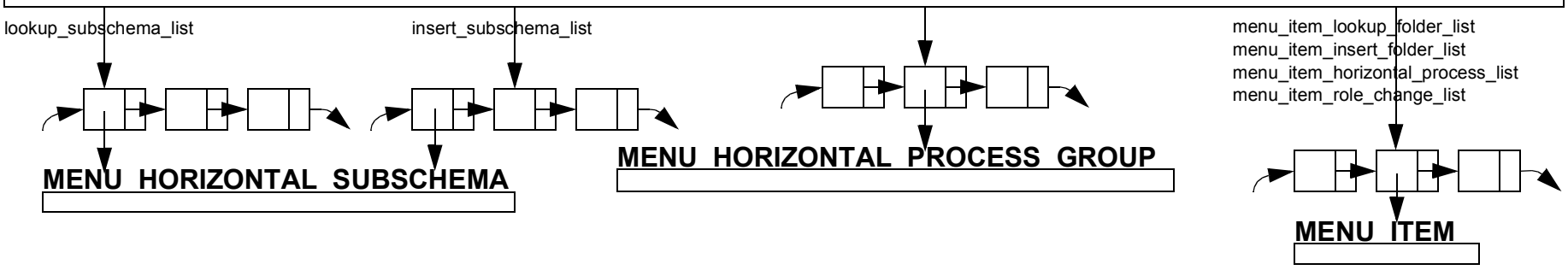


MENU_HORIZONTAL (1)

```
/* Usage */
MENU_HORIZONTAL *
   menu_horizontal_new(
      char *application_name,
      char *session_key,
      char *login_name,
      char *role_name,
      LIST *folder_menu_count_list,
      LIST *role_folder_lookup_list,
      LIST *role_folder_insert_list,
      LIST *role_process_list,
      LIST *appaserver_user_role_name_list,
      LIST *role_folder_subschema_name_list,
      LIST *role_folder_subschema_missing_folder_name_list,
      LIST *role_process_group_name_list,
      LIST *role_process_missing_group_name_list );

/* Process */
MENU_HORIZONTAL *menu_horizontal_calloc( void );

LIST *lookup_subschema_list = list_new();

for char *subschema_name in
   role_folder_subschema_name_list
{
   list_set(
      lookup_subschema_list,
      menu_horizontal_subschema_new(
         application_name,
         session_key,
         login_name,
         role_name,
         folder_menu_count_list,
         role_folder_lookup_list
            /* role_folder_list */,
         subschema_name,
         APPASERVER_LOOKUP_STATE ) );
}

LIST *insert_subschema_list = list_new();

for char *subschema_name in
   role_folder_subschema_name_list
{
   list_set(
      insert_subschema_list,
      menu_horizontal_subschema_new(
         application_name,
         session_key,
         login_name,
         role_name,
         (LIST *)0 /* folder_menu_count_list */,
         role_folder_insert_list()
            /* role_folder_list */,
         subschema_name,
         APPASERVER_INSERT_STATE ) );
}

LIST *menu_item_lookup_folder_list = list_new();

for char *folder_name in
   role_folder_subschema_missing_folder_name_list
{
   FOLDER_MENU_COUNT *folder_menu_count =
      folder_menu_count_seek(
         folder_name,
         folder_menu_count_list );

   list_set(
      menu_item_lookup_folder_list,
      menu_item_horizontal_folder_new(
         application_name,
         session_key,
         login_name,
         role_name,
         folder_name,
         (folder_menu_count)
            ? folder_menu_count->display
            : (char *)0,
         APPASERVER_LOOKUP_STATE );
}

LIST *menu_item_insert_folder_list = list_new();

for char *folder_name in
   role_folder_subschema_missing_folder_name_list
{
   if ( role_folder_insert_exists(
      folder_name,
      role_folder_insert_list ) )
   {
      list_set(
         menu_item_insert_folder_list,
         menu_item_horizontal_folder_new(
            application_name,
            session_key,
            login_name,
            role_name,
            folder_name,
            (char *)0 /* folder_menu_count_display */,
            APPASERVER_INSERT_STATE );
   }
}

LIST *menu_horizontal_process_group_list = list_new();

for char *group_name in
   role_process_group_name_list
{
   list_set(
      menu_horizontal_process_group_list,
      menu_horizontal_process_group_new(
         application_name,
         session_key,
         login_name,
         role_name ,
         group_name,
         role_process_list ) );
}
```

MENU_HORIZONTAL (2)

```
/* Process (continued) */
LIST *menu_item_horizontal_process_list = list_new();

for char *process_name in
    role_process_missing_group_name_list
{
    list_set(
        menu_item_horizontal_process_list,
        menu_item_horizontal_process_new(
            application_name,
            session_key,
            login_name,
            role_name,
            process_name ) );
}

if ( list_length( appaserver_user_role_name_list ) > 1 )
{
    LIST *menu_item_role_change_list = list_new();

    LIST *menu_item_role_name_list(
        char *role_name,
        LIST *appaserver_user_role_name_list );

    for char *local_role_name in
        menu_item_role_name_list()
    {
        list_set(
            menu_item_role_change_list,
            menu_item_horizontal_role_change_new(
                application_name,
                session_key,
                login_name,
                local_role_name ) );
    }
}

/* Note: frees each ->html */
char *menu_horizontal_html(
    login_name,
    role_name,
    lookup_subschema_list,
    insert_subschema_list,
    menu_item_lookup_folder_list,
    menu_item_insert_folder_list,
    menu_horizontal_process_group_list,
    menu_item_horizontal_process_list,
    menu_item_role_change_list );

/* Usage */
char *menu_horizontal_heading_span_tag(
    char *MENU_ITEM_HORIZONTAL_CLASS_NAME,
    char *heading_string );

/* Process */
char *string_initial_capital(
    buffer[],
    heading_string );

sprintf( static tag[],
    "<li><a><span class=%s>%s</span></a>",
    MENU_ITEM_HORIZONTAL_CLASS_NAME,
    string_initial_capital() );

/* Attributes */
LIST *lookup_subschema_list;
LIST *insert_subschema_list;
LIST *menu_item_lookup_folder_list;
LIST *menu_item_insert_folder_list;
LIST *menu_horizontal_process_group_list;
LIST *menu_item_horizontal_process_list;
LIST *menu_item_role_name_list;
LIST *menu_item_role_change_list;
char *html;
```



MENU_HORIZONTAL (3)

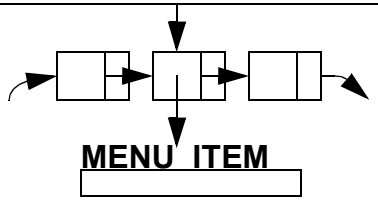
```
/* Usage */
/* Note: frees each ->html */
char *menu_horizontal_html(
    char *login_name,
    char *role_name,
    LIST *lookup_subschema_list,
    LIST *insert_subschema_list,
    LIST *menu_item_lookup_folder_list,
    LIST *menu_item_insert_folder_list,
    LIST *menu_horizontal_process_group_list,
    LIST *menu_item_horizontal_process_list,
    LIST *menu_item_role_change_list );

/* Process */
char html[];
char *ptr = html;

ptr += sprintf( ptr,
    "%s\n",
    MENU_UNORDERED_LIST_TAG );

if (    list_length( lookup_subschema_list )
||      list_length( menu_item_lookup_folder_list ) )
{
    char *menu_horizontal_heading_span_tag(
        MENU_ITEM_HORIZONTAL_CLASS_NAME,
        "Lookup" /* heading_string */ );

    ptr += sprintf( ptr,
        "%s\n<ul>\n",
        menu_horizontal_heading_span_tag() );
}

for MENU_HORIZONTAL_SUBSCHEMA *subschema in
    lookup_subschema_list
{
    ptr += sprintf( ptr,
        "%s\n",
        subschema->html );

    free( subschema->html );
}

for MENU_ITEM *menu_item in
    menu_item_lookup_folder_list
{
    ptr += sprintf( ptr,
        "\t%s\n",
        menu_item->html );

    free( menu_item->html );
}

if (    list_length( lookup_subschema_list )
||      list_length( menu_item_lookup_folder_list ) )
{
    ptr += sprintf( ptr, "</ul>\n" );
}

if (    list_length( insert_subschema_list )
||      list_length( menu_item_insert_folder_list ) )
{
    char *menu_horizontal_heading_span_tag(
        MENU_ITEM_HORIZONTAL_CLASS_NAME,
        "Insert" /* heading_string */ );

    ptr += sprintf( ptr,
        "%s\n<ul>\n",
        menu_horizontal_heading_span_tag() );
}

for MENU_HORIZONTAL_SUBSCHEMA *subschema in
    insert_subschema_list
{
    ptr += sprintf( ptr,
        "%s\n",
        subschema->html );

    free( subschema->html );
}

for MENU_ITEM *menu_item in
    menu_item_insert_folder_list
{
    ptr += sprintf( ptr,
        "\t%s\n",
        menu_item->html );

    free( menu_item->html );
}

if (    list_length( insert_subschema_list )
||      list_length( menu_item_insert_folder_list ) )
{
    ptr += sprintf( ptr, "</ul>\n" );
}
```

MENU_HORIZONTAL (4)

```
/* menu_horizontal_html() continued */
for MENU_HORIZONTAL_PROCESS_GROUP *
    process_group in
        menu_horizontal_process_group_list
{
    ptr += sprintf( ptr,
        "%s\n",
        process_group->html );

    free( process_group->html );
}

if ( list_rewind( menu_item_horizontal_process_list ) )
{
    char *menu_horizontal_heading_span_tag(
        MENU_ITEM_HORIZONTAL_CLASS_NAME,
        "Process" /* heading_string */ );

    ptr += sprintf( ptr,
        "%s\n<ul>\n",
        menu_horizontal_heading_span_tag() );
}

for MENU_ITEM *menu_item in
    menu_item_horizontal_process_list
{
    ptr += sprintf( ptr,
        "%s\n",
        menu_item->html );

    free( menu_item->html );
}

if ( list_length( menu_item_horizontal_process_list ) )
{
    ptr += sprintf( ptr, "</ul>\n" );
}

if ( list_rewind( menu_item_role_change_list ) )
{
    char *menu_horizontal_heading_span_tag(
        MENU_ITEM_HORIZONTAL_CLASS_NAME,
        "Role" /* heading_string */ );

    ptr += sprintf( ptr,
        "%s\n<ul>\n",
        menu_horizontal_heading_span_tag() );
}

for MENU_ITEM *menu_item in
    menu_item_role_change_list
{
    ptr += sprintf( ptr,
        "%s\n",
        menu_item->html );

    free( menu_item->html );
}

if ( list_length( menu_item_role_change_list ) )
{
    ptr += sprintf( ptr, "</ul>\n" );
}

char *string_initial_capital(
    buffer[],
    role_name );

ptr += sprintf( ptr,
    "<li><a>" );

ptr += sprintf( ptr,
    "<label style=color:black>%s</label></a>\n",
    string_initial_capital() );

char *string_initial_capital(
    buffer[],
    login_name );

ptr += sprintf( ptr,
    "<li><a>" );

ptr += sprintf( ptr,
    "<label style=color:black>%s</label></a>\n",
    string_initial_capital() );

sprintf( ptr, "</ul>" );
```

MENU_VERTICAL_SUBSCHEMA

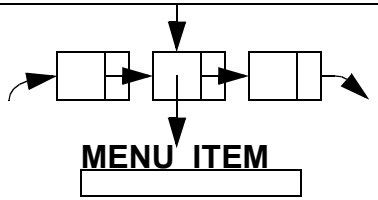
```
/* Usage */
MENU_VERTICAL_SUBSCHEMA *
    menu_vertical_subschema_new(
        char *application_name,
        char *session_key,
        char *login_name,
        char *role_name,
        LIST *folder_menu_count_list(),
        LIST *role_folder_lookup_list(),
        LIST *role_folder_insert_list(),
        char *subschema_name );

/* Process */
MENU_VERTICAL_SUBSCHEMA *
    menu_vertical_subschema_calloc(
        void );

LIST *role_folder_subschema_folder_name_list(
        subschema_name,
        role_folder_lookup_list
            /* role_folder_list */ );

if ( !list_rewind(
        role_folder_subschema_folder_name_list() )
            return (MENU_VERTICAL_SUBSCHEMA *)0;

LIST *menu_item_list = list_new();

for  folder_name in
    role_folder_subschema_folder_name_list()
{
    boolean role_folder_insert_exists(
        folder_name,
        role_folder_insert_list );

    FOLDER_MENU_COUNT *folder_menu_count =
        folder_menu_count_seek(
            folder_name,
            folder_menu_count_list );

    list_set(
        menu_item_list,
        menu_item_vertical_folder_new(
            application_name,
            session_key,
            login_name,
            role_name,
            folder_name,
            (folder_menu_count)
                ? folder_menu_count->display
                : (char *)0,
            role_folder_insert_exists() ) );
}

char *menu_item_list_vertical_html(
    subschema_name /* heading_string */,
    menu_item_list );

free( menu_item_list );

/* Attributes */
LIST *role_folder_subschema_folder_name_list;
LIST *menu_item_list;
char *html;
```



MENU_HORIZONTAL_SUBSCHEMA

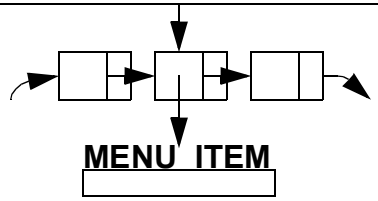
```
/* Usage */
MENU_HORIZONTAL_SUBSCHEMA *
   menu_horizontal_subschema_new(
      char *application_name,
      char *session_key,
      char *login_name,
      char *role_name,
      LIST *folder_menu_count_list(),
      LIST *role_folder_list
               /* role_folder[_lookup|insert]_list */,
      char *subschema_name,
      char *state );

/* Process */
MENU_HORIZONTAL_SUBSCHEMA *
   menu_horizontal_subschema_calloc(
      void );

LIST *role_folder_subschema_folder_name_list(
      subschema_name,
      role_folder_list );
```

```
if ( !list_rewind(
      role_folder_subschema_folder_name_list() )
         return (MENU_HORIZONTAL_SUBSCHEMA *)0;

LIST *menu_item_list = list_new();

for  folder_name in
   role_folder_subschema_folder_name_list()
{
   FOLDER_MENU_COUNT *folder_menu_count =
      folder_menu_count_seek(
         folder_name,
         folder_menu_count_list );

   list_set(
      menu_item_list,
      menu_item_horizontal_folder_new(
         application_name,
         session_key,
         login_name,
         role_name,
         folder_name,
         state,
         (folder_menu_count)
            ? folder_menu_count->display
            : (char *)0 ) );
}

char *menu_item_heading_span_tag(
   MENU_ITEM_HORIZONTAL_CLASS_NAME,
   subschema_name /* heading_string */ );

char *menu_item_list_horizontal_html(
   menu_item_heading_span_tag(),
   menu_item_list );

list_free( menu_item_list );

/* Attribute */
char *subschema_name;
LIST *role_folder_subschema_folder_name_list;
LIST *menu_item_list;
char *menu_item_heading_span_tag;
char *html;
```



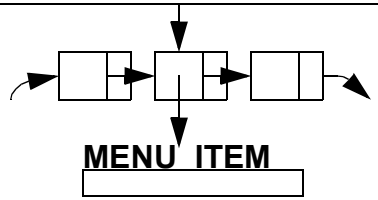
MENU_VERTICAL_PROCESS_GROUP

```
/* Usage */
MENU_VERTICAL_PROCESS_GROUP *
   menu_vertical_process_group_new(
      char *application_name,
      char *session_key,
      char *login_name,
      char *role_name,
      char *group_name,
      LIST *role_process_list );

/* Process */
MENU_VERTICAL_PROCESS_GROUP *
   menu_vertical_process_group_calloc(
      void );

LIST *role_process_name_list(
      group_name,
      role_process_list );

if ( !list_rewind( role_process_name_list() )
      return (MENU_VERTICAL_PROCESS_GROUP *)0;

LIST *menu_item_list = list_new()

for process_name in
   role_process_name_list()
{
   list_set(
      menu_item_list,
      menu_item_vertical_process_new(
         application_name,
         session_key,
         login_name,
         role_name,
         process_name ) );
}

char *html =
   menu_item_list_vertical_html(
      group_name /* heading_string */,
      menu_item_list );

list_free( menu_item_list );

/* Attributes */
LIST *role_process_name_list;
LIST *menu_item_list;
char *html;
```



MENU_HORIZONTAL_PROCESS_GROUP

```
/* Usage */
MENU_HORIZONTAL_PROCESS_GROUP *
    menu_horizontal_process_group_new(
        char *application_name,
        char *session_key,
        char *login_name,
        char *role_name,
        char *group_name,
        LIST *role_process_list );

/* Process */
MENU_HORIZONTAL_PROCESS_GROUP *
    menu_horizontal_process_group_calloc(
        void );

LIST *role_process_name_list(
    group_name,
    role_process_list );

if ( !list_rewind( role_process_name_list() ) return null;

LIST *menu_item_list = list_new()

for char *process_name in role_process_name_list()
{
    list_set(
        menu_item_list,
        menu_item_horizontal_process_new(
            application_name,
            session_key,
            login_name,
            role_name,
            process_name ) );
}

char *menu_item_heading_span_tag(
    MENU_ITEM_HORIZONTAL_CLASS_NAME,
    group_name /* heading_string */ );

char *menu_item_list_horizontal_html(
    menu_item_heading_span_tag(),
    menu_item_list );

list_free( menu_item_list );

/* Attribute */
LIST *role_process_name_list;
LIST *menu_item_list;
char *menu_item_heading_span_tag;
char *html;
```



MENU_ITEM (1)

```
/* Usage */
MENU_ITEM *menu_item_vertical_folder_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *folder_menu_count_display,
    boolean role_folder_insert_exists() );

/* Process */
MENU_ITEM *menu_item_calloc( void );

char *post_choose_folder_action_string(
    POST_CHOOSE_FOLDER_EXECUTABLE,
    application_name,
    session_key,
    login_name,
    role_name,
    APPASERVER_LOOKUP_STATE,
    folder_name );

char *lookup_anchor_tag =
    char *strdup(
        menu_item_anchor_tag(
            FRAMESET_MENU_FRAME /* target_frame */,
            post_choose_folder_action_string() );

if ( role_folder_insert_exists )
{
    char *post_choose_folder_action_string(
        POST_CHOOSE_FOLDER_EXECUTABLE,
        application_name,
        session_key,
        login_name,
        role_name,
        APPASERVER_INSERT_STATE,
        folder_name );

    char *insert_anchor_tag =
        menu_item_anchor_tag(
            FRAMESET_MENU_FRAME /* target_frame */,
            post_choose_folder_action_string() );
}

char *menu_item_vertical_folder_html(
    folder_name,
    folder_menu_count_display,
    lookup_anchor_tag,
    insert_anchor_tag );

free( lookup_anchor_tag );

/* Usage */
char *menu_item_vertical_folder_html(
    char *folder_name,
    char *folder_menu_count_display,
    char *lookup_anchor_tag,
    char *insert_anchor_tag );

/* Process */
char html[];
char *ptr = html;

char *menu_item_span_tag(
    MENU_ITEM_VERTICAL_CLASS_NAME,
    folder_name /* item */ );

char *menu_item_vertical_html(
    menu_item_span_tag(),
    lookup_anchor_tag,
    1 /* table_row_boolean */ );

ptr += sprintf( ptr,
    "%s",
    menu_item_vertical_html() );

if ( folder_menu_count_display )
{
    ptr += sprintf( ptr,
        "<td align=right>%s",
        folder_menu_count_display );
}

if ( insert_anchor_tag )
{
    char *menu_item_span_tag(
        MENU_ITEM_VERTICAL_CLASS_NAME,
        "new" /* item */ );

    char *menu_item_vertical_html(
        menu_item_span_tag(),
        insert_anchor_tag,
        0 /* not table_row_boolean */ );

    ptr += sprintf( ptr,
        "%s",
        menu_item_vertical_html() );
}

sprintf( ptr, "\n" );
```

MENU_ITEM (2)

```
/* Usage */
MENU_ITEM *menu_item_vertical_process_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *process_name );

/* Process */
MENU_ITEM *menu_item_calloc();

char *menu_item_span_tag(
    MENU_ITEM_VERTICAL_CLASS_NAME,
    process_name /* item */ );

char *post_choose_process_action_string(
    POST_CHOOSE_PROCESS_EXECUTABLE,
    application_name,
    session_key,
    login_name,
    role_name );

char *menu_item_href_string(
    process_name /* item */,
    post_choose_process_action_string() );

char *menu_item_anchor_tag(
    FRAMESET_MENU_FRAME /* target_frame */,
    menu_item_href_string() );

char *strdup(
    char *menu_item_vertical_html(
        menu_item_span_tag(),
        menu_item_anchor_tag(),
        1 /* table_row_boolean */ ) );

/* Usage */
char *menu_item_vertical_html(
    char *menu_item_span_tag(),
    char *anchor_tag,
    boolean table_row_boolean );

/* Process */
static char html[];
char *ptr = html;

if ( table_row_boolean ) ptr += sprintf( ptr, "<tr>" );

sprintf(
    ptr,
    "<td>%s%s</a>",
    anchor_tag,
    menu_item_span_tag );

/* Usage */
MENU_ITEM *menu_item_vertical_role_change_new(
    char *application_name,
    char *session_key,
    char *login_name,
    LIST *menu_item_role_name_list() );

/* Process */
MENU_ITEM *menu_item_calloc();

char *post_choose_role_action_string(
    POST_CHOOSE_ROLE_EXECUTABLE,
    application_name,
    session_key,
    login_name );

FORM_CHOOSE_ROLE *form_choose_role =
    form_choose_role_new(
        menu_item_role_name_list(),
        post_choose_role_action_string(),
        FRAMESET_MENU_FRAME /* target_frame */,
        FORM_CHOOSE_ROLE_NAME,
        CHOOSE_ROLE_DROP_DOWN_NAME );

char *html = form_choose_role->html;

/* Usage */
char *menu_item_list_vertical_html(
    char *heading_name,
    LIST *menu_item_list );

/* Process */
char html[];
char *ptr = html;

if ( !list_rewind( menu_item_list ) ) return (char *)0;

ptr += sprintf( ptr,
    "<tr><th colspan=3>%s\n"
    string_initial_capital(
        buffer[],
        heading_name ) );

for MENU_ITEM *menu_item in menu_item_list
{
    ptr += sprintf( ptr,
        "%s\n",
        menu_item->html );
}
```

MENU_ITEM (3)

```
/* Usage */
MENU_ITEM *menu_item_horizontal_folder_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *state,
    char *folder_menu_count_display );

/* Process */
MENU_ITEM *menu_item_calloc();

char *menu_item_horizontal_folder_span_tag(
    MENU_ITEM_HORIZONTAL_CLASS_NAME,
    folder_name,
    folder_menu_count_display );

char *post_choose_folder_action_string(
    POST_CHOOSE_FOLDER_EXECUTABLE,
    application_name,
    session_key,
    login_name,
    role_name,
    state,
    folder_name);

char *menu_item_anchor_tag(
    FRAMESET_PROMPT_FRAME /* target_frame */,
    post_choose_folder_action_string() );

char *menu_item_horizontal_html(
    menu_item_horizontal_folder_span_tag(),
    menu_item_anchor_tag() );

/* Usage */
MENU_ITEM *menu_item_horizontal_process_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *process_name );

/* Process */
MENU_ITEM *menu_item_calloc();

char *menu_item_span_tag(
    MENU_ITEM_HORIZONTAL_CLASS_NAME,
    process_name /* item */ );

char *post_choose_process_action_string(
    POST_CHOOSE_PROCESS_EXECUTABLE,
    application_name,
    session_key,
    login_name,
    role_name );

char *menu_item_href_string(
    process_name /* item */,
    post_choose_process_action_string() );

char *menu_item_anchor_tag(
    FRAMESET_PROMPT_FRAME /* target_frame */,
    menu_item_href_string() );

char *menu_item_horizontal_html(
    menu_item_span_tag(),
    menu_item_anchor_tag() );

/* Usage */
MENU_ITEM *menu_item_horizontal_role_change_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name );

/* Process */
MENU_ITEM *menu_item_calloc();

char *menu_item_span_tag(
    MENU_ITEM_HORIZONTAL_CLASS_NAME,
    role_name /* item */ );

char *post_choose_role_action_string(
    POST_CHOOSE_ROLE_EXECUTABLE,
    application_name,
    session_key,
    login_name );

char *menu_item_href_string(
    role_name /* item */,
    post_choose_role_action_string() );

char *menu_item_anchor_tag(
    FRAMESET_PROMPT_FRAME /* target_frame */,
    menu_item_href_string() );

char *menu_item_horizontal_html(
    menu_item_span_tag(),
    menu_item_anchor_tag() );

/* Usage */
char *menu_item_horizontal_folder_span_tag(
    char *MENU_ITEM_HORIZONTAL_CLASS_NAME,
    char *folder_name,
    char *folder_menu_count_display );

/* Process */
static char span_tag[];
char *ptr = span_tag;

ptr += sprintf( ptr,
    "<span class=%s>%s",
    MENU_ITEM_HORIZONTAL_CLASS_NAME,
    string_initial_capital(
        buffer[],
        folder_name ) );

if ( folder_menu_count_display )
{
    ptr += sprintf( ptr,
        " %s",
        folder_menu_count_display );
}

sprintf( ptr, "</span>" );
```

MENU_ITEM (4)

```
/* Usage */
char *menu_item_horizontal_html(
    char *menu_item_span_tag(),
    char *menu_item_anchor_tag() );

/* Process */
sprintf( html[],
    "<li>%s%s</a>",
    menu_item_anchor_tag,
    menu_item_span_tag );

/* Usage */
/* Note: frees each ->html */
char *menu_item_list_horizontal_html(
    char *menu_item_heading_span_tag(),
    LIST *menu_item_list );

/* Process */
char html[];
char *ptr = html;

if ( !list_rewind( menu_item_list ) ) return (char *)0;

ptr += sprintf( ptr,
    "\t%s\n\t<ul>\n",
    menu_item_heading_span_tag );

for MENU_ITEM *menu_item in menu_item_list
{
    ptr += sprintf( ptr,
        "\t\t%s",
        menu_item->html );

    free( menu_item->html );
}

ptr += sprintf( ptr, "\t</ul>" );

/* Usage */
char *menu_item_heading_span_tag(
    char *class_name,
    char *heading_string );

/* Process */
sprintf( static span_tag[],
    "<li><a>%s</a>",
    char *menu_item_span_tag(
        class_name,
        heading_string /* item */ ) );

/* Usage */
char *menu_item_span_tag(
    char *class_name,
    char *item );

/* Process */
sprintf( static span_tag[],
    "<span class=%s>%s</span>",
    class_name,
    string_initial_capital(
        buffer[],
        item ) );

/* Usage */
char *menu_item_href_string(
    char *item,
    char *action_string );

/* Process */
sprintf(
    static href_string[],
    "%s+%s",
    action_string,
    item );

/* Usage */
char *menu_item_anchor_tag(
    char *target_frame,
    char *menu_item_href_string() );

/* Process */
snprintf(
    static anchor_tag[],
    sizeof ( anchor_tag ),
    "<a href=\"%s\" target=%s>"
    menu_item_href_string,
    target_frame );
/*  JAVASCRIPT_WAIT_START ); */

/* Usage */
LIST *menu_item_role_name_list(
    char *role_name,
    LIST *appaserver_user_role_name_list() );

/* Process */
LIST *list = list_new();

for char *appaserver_user_role_name in
    appaserver_user_role_name_list
{
    if ( strcmp(
        appaserver_user_role_name,
        role_name ) != 0 )
    {
        list_set(
            list,
            appaserver_user_role_name );
    }
}

/* Attributes */
char *span_tag;
char *action_string;
char *href_string;
char *anchor_tag;
char *html;
```

RECALL

```
/* Usage */
boolean recall_ignore_boolean(
    const char *APPASERVER_EXECUTE_YN,
    char *widget_name,
    enum widget_type widget_type,
    boolean recall_boolean,
    boolean multi_drop_down_boolean );

/* Process */
if ( !widget_name ) return 1;

if ( !recall_boolean ) return 1;

int string_strcmp(
    widget_name,
    APPASERVER_EXECUTE_YN );

if ( string_strcmp() == 0 ) return 1;

if (    multi_drop_down_boolean
&&   widget_type != multi_drop_down )
{
    return 1;
}

if (    !multi_drop_down_boolean
&&   widget_type == multi_drop_down )
{
    return 1;
}

return 0;

/* Usage */
static char *recall_widget_component_string(
    const char RECALL_DELIMITER,
    WIDGET_CONTAINER *widget_container,
    boolean got_one );

/* Process */
static char component_string[];
char *ptr = component_string;

if ( got_one )
{
    ptr +=
        sprintf(
            ptr,
            "%c",
            RECALL_DELIMITER );
}

char *recall_widget_name(
    widget_container );

ptr +=
    sprintf(
        ptr,
        "%s",
        recall_widget_name() );

/* Usage */
char *recall_widget_name(
     WIDGET_CONTAINER *widget_container );

/* Process */
if ( widget_container->widget_type == upload )
     char *widget_container->upload->recall_widget_name;
else
     char *widget_container->widget_name;
```

RECALL_SAVE (1)

```
/* Usage */
RECALL_SAVE *recall_save_new(
    char *state,
    char *form_name,
    char *folder_name,
    LIST *widget_container_list,
    boolean application_ssl_support_boolean );

/* Process */
RECALL_SAVE *recall_save_calloc( void );

char *recall_save_cookie_key(
    char *RECALL_PROCESS_COOKIE_KEY,
    char *folder_name,
    char *state );

char *recall_save_keystrokes_javascript(
    RECALL_DELIMITER,
    WIDGET_MULTI_LEFT_RIGHT_DELIMITER,
    form_name,
    widget_container_list,
    application_ssl_support_boolean,
    recall_save_cookie_key() );

char *recall_save_cookie_multi_key(
    char *folder_name,
    char *state );

char *recall_save_keystrokes_multi_javascript(
    RECALL_DELIMITER,
    WIDGET_MULTI_LEFT_RIGHT_DELIMITER ,
    form_name,
    widget_container_list,
    application_ssl_support_boolean,
    recall_save_cookie_multi_key() );

char *javascript =
    char *recall_load_javascript(
        recall_save_keystrokes_javascript(),
        recall_save_keystrokes_multi_javascript() );

/* Usage */
char *recall_save_keystrokes_javascript(
    const char RECALL_DELIMITER,
    const char WIDGET_MULTI_LEFT_RIGHT_DELIMITER,
    char *form_name,
    LIST *widget_container_list,
    boolean appliction_ssl_support_boolean,
    char *recall_save_cookie_key() );

/* Process */
char javascript[];
char *ptr = javascript;
boolean got_one = 0;

char *recall_save_keystrokes_open(
    form_name,
    application_ssl_support_boolean,
    recall_save_cookie_key );

ptr +=
    sprintf(
        ptr,
        "%s",
        recall_save_keystrokes_open() );

for WIDGET_CONTAINER *widget_container in
    widget_container_list
{
    boolean recall_ignore_boolean(
        APPASERVER_EXECUTE_YN,
        widget_container->widget_name,
        widget_container->widget_type,
        widget_container->recall_boolean,
        0 /* not multi_drop_down_boolean */ );

    if ( recall_ignore_boolean() ) continue;

    static char *recall_widget_component_string(
        RECALL_DELIMITER,
        widget_container,
        got_one );

    ptr += sprintf( ptr, "%s", recall_widget_component_string() );

    got_one = 1;
}

if ( !got_one ) return (char *)0;

char *recall_save_keystrokes_close(
    char RECALL_DELIMITER,
    char WIDGET_MULTI_LEFT_RIGHT_DELIMITER );

ptr +=
    sprintf(
        ptr,
        "%s",
        recall_save_keystrokes_close() );

/* Attributes */
char *cookie_key;
char *keystrokes_javascript;
char *cookie_multi_key;
char *keystrokes_multi_javascript;
char *javascript;
```

RECALL_SAVE (2)

```
/* Usage */
char *recall_save_keystrokes_multi_javascript(
    const char RECALL_DELIMITER,
    const char WIDGET_MULTI_LEFT_RIGHT_DELIMITER ,
    char *form_name,
    LIST *widget_container_list,
    boolean application_ssl_support_boolean,
    char *recall_save_cookie_multi_key() );

/* Process */
char javascript[];
char *ptr = javascript;
boolean got_one = 0;

char *recall_save_keystrokes_multi_open(
    form_name,
    application_ssl_support_boolean,
    recall_save_cookie_multi_key );

ptr +=
  sprintf(
    ptr,
    "%s",
    recall_save_keystrokes_multi_open() );

for WIDGET_CONTAINER *widget_container in
  widget_container_list
{
  boolean recall_ignore_boolean(
    APPASERVER_EXECUTE_YN,
    widget_container->widget_name,
    widget_container->widget_type,
    widget_container->recall_boolean,
    1 /* multi_drop_down_boolean */ );

  if ( recall_ignore_boolean() ) continue;

  static char *recall_widget_component_string(
    RECALL_DELIMITER,
    widget_container,
    got_one );

  ptr += sprintf( ptr, "%s", recall_widget_component_string() );

  got_one = 1;
}

if ( !got_one ) return (char *)0;

char *recall_save_keystrokes_multi_close(
    RECALL_DELIMITER,
    WIDGET_MULTI_LEFT_RIGHT_DELIMITER );

ptr += sprintf( ptr,
    "%s",
    recall_save_keystrokes_multi_close() );
```

```
/* Usage */
char *recall_save_keystrokes_open(
    char *form_name,
    boolean application_ssl_support_boolean,
    char *recall_save_cookie_key );

/* Process */
char *recall_save_samesite_string(
    application_ssl_support_boolean );

char *recall_save_secure_string(
    application_ssl_support_boolean );

sprintf( static open[],
    "keystrokes_save(%s,'%s','%s','%s','",
    form_name,
    recall_save_cookie_key,
    recall_save_samesite_string(),
    recall_save_secure_string() );

/* Usage */
char *recall_save_keystrokes_close(
    char RECALL_DELIMITER,
    char WIDGET_MULTI_LEFT_RIGHT_DELIMITER );

/* Process */
sprintf( static close[],
    "','%c','%c'",
    RECALL_DELIMITER,
    WIDGET_MULTI_LEFT_RIGHT_DELIMITER );

/* Usage */
char *recall_save_keystrokes_multi_open(
    char *form_name,
    boolean application_ssl_support_boolean,
    char *recall_save_cookie_multi_key );

/* Process */
char *recall_save_samesite_string(
    application_ssl_support_boolean );

char *recall_save_secure_string(
    application_ssl_support_boolean );

sprintf( static open[],
    "keystrokes_multi_save(%s,'%s','%s','%s','",
    form_name,
    recall_save_cookie_multi_key,
    recall_save_samesite_string(),
    recall_save_secure_string() );

/* Usage */
char *recall_save_keystrokes_multi_close(
    char RECALL_DELIMITER,
    char WIDGET_MULTI_LEFT_RIGHT_DELIMITER );

/* Process */
sprintf( static close[],
    "','%c','%c','%c')",
    RECALL_DELIMITER,
    WIDGET_MULTI_LEFT_RIGHT_DELIMITER,
    RECALL_DELIMITER
        /* multi_select_remember_delimiter */ );

/* Usage */
char *recall_save_samesite_string(
    boolean application_ssl_support_boolean );

/* Process */
if ( application_ssl_support_boolean )
    return "strict";
else
    return "";

/* Usage */
char *recall_save_secure_string(
    boolean application_ssl_support_boolean );

/* Process */
if ( application_ssl_support_boolean )
    return "secure";
else
    return "";
```

RECALL_LOAD (1)

```
/* Usage */
RECALL_LOAD *recall_load_new(
    char *state,
    char *form_name,
    char *folder_name,
    LIST *widget_container_list );

/* Process */
RECALL_LOAD *recall_load_calloc( void );

char *recall_save_cookie_key(
    RECALL_PROCESS_COOKIE_KEY,
    folder_name,
    state );

char *recall_load_keystrokes_javascript(
    RECALL_DELIMITER,
    WIDGET_MULTI_LEFT_RIGHT_DELIMITER,
    form_name,
    widget_container_list,
    recall_save_cookie_key() );

char *recall_save_cookie_multi_key(
    folder_name,
    state );

char *recall_load_keystrokes_multi_javascript(
    RECALL_DELIMITER,
    WIDGET_MULTI_LEFT_RIGHT_DELIMITER,
    form_name,
    widget_container_list,
    recall_save_cookie_multi_key() );

char *recall_load_javascript(
    char *recall_load_keystrokes_javascript(),
    char *recall_load_keystrokes_multi_javascript() );

/* Usage */
char *recall_load_keystrokes_javascript(
    const char RECALL_DELIMITER,
    const char WIDGET_MULTI_LEFT_RIGHT_DELIMITER,
    char *form_name,
    LIST *widget_container_list,
    char *recall_save_cookie_key() );

/* Process */
char javascript[];
char *ptr = javascript;
boolean got_one = 0;

char *recall_load_keystrokes_open(
    form_name,
    recall_save_cookie_multi_key );

ptr +=
  sprintf(
    ptr,
    "%s",
    recall_load_keystrokes_open() );

for WIDGET_CONTAINER *widget_container in
  widget_container_list
{
  boolean recall_ignore_boolean(
    APPASERVER_EXECUTE_YN,
    widget_container->widget_name,
    widget_container->widget_type,
    widget_container->recall_boolean,
    0 /* not multi_drop_down_boolean */ );

  if ( recall_ignore_boolean() ) continue;

  static char *recall_widget_component_string(
    RECALL_DELIMITER,
    widget_container,
    got_one );

  ptr += sprintf( ptr, "%s", recall_widget_component_string() );

  got_one = 1;
}

if ( !got_one ) return (char *)0;

char *recall_load_keystrokes_close(
    RECALL_DELIMITER,
    WIDGET_MULTI_LEFT_RIGHT_DELIMITER );

ptr += sprintf( ptr,
    "%s",
    recall_load_keystrokes_close() );

/* Attributes */
char *recall_save_cookie_key;
char *keystrokes_javascript;
char *recall_save_cookie_multi_key;
char *keystrokes_multi_javascript;
char *javascript;
```

RECALL_LOAD (2)

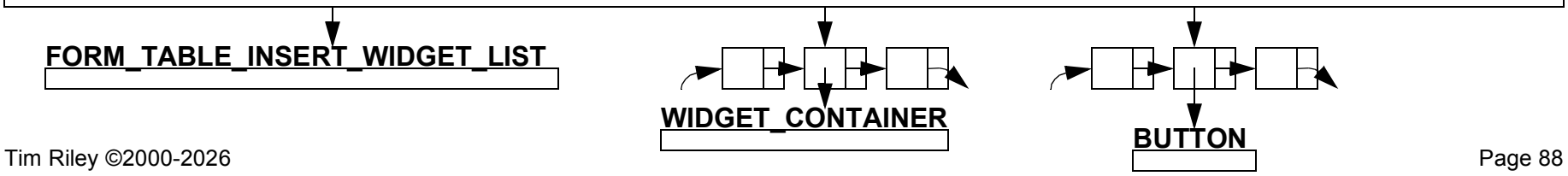
```
/* Usage */
char *recall_load_keystrokes_multi_javascript(
     const char RECALL_DELIMITER,
     const char WIDGET_MULTI_LEFT_RIGHT_DELIMITER,
     char *form_name,
     LIST *widget_container_list,
     char *recall_save_cookie_multi_key() );

/* Process */
char javascript[];
char *ptr = javascript;
boolean got_one = 0;

char *recall_load_keystrokes_multi_open(
     form_name,
     recall_save_cookie_multi_key );

ptr +=
   sprintf(
      ptr,
      "%s",
      recall_load_keystrokes_multi_open() );

for WIDGET_CONTAINER *widget_container in
   widget_container_list
{
   boolean recall_ignore_boolean(
      APPASERVER_EXECUTE_YN,
      widget_container->widget_name,
      widget_container->widget_type,
      widget_container->recall_boolean,
      1 /* multi_drop_down_boolean */ );

   if ( recall_ignore_boolean() ) continue;

   static char *recall_widget_component_string(
      RECALL_DELIMITER,
      widget_container,
      got_one );

   ptr += sprintf( ptr, "%s", recall_widget_component_string() );

   got_one = 1;
}

if ( !got_one ) return (char *)0;

char *recall_load_keystrokes_multi_close(
     RECALL_DELIMITER,
     WIDGET_MULTI_LEFT_RIGHT_DELIMITER );

ptr += sprintf( ptr,
     "%s",
     recall_load_keystrokes_multi_close() );

/* Usage */
char *recall_load_keystrokes_open(
     char *form_name,
     char *recall_save_cookie_key );

/* Process */
sprintf( static open[],
     "keystrokes_onload(%s,'%s','",
     form_name,
     recall_save_cookie_key );

/* Usage */
char *recall_load_keystrokes_close(
     char RECALL_DELIMITER,
     char WIDGET_MULTI_LEFT_RIGHT_DELIMITER );

/* Process */
sprintf( static close[],
     "','%c','%c')",
     RECALL_DELIMITER,
     WIDGET_MULTI_LEFT_RIGHT_DELIMITER );

/* Usage */
char *recall_load_keystrokes_multi_open(
     char *form_name,
     char *recall_save_cookie_multi_key );

/* Process */
sprintf( static open[],
     "keystrokes_multi_onload(%s,'%s','",
     form_name,
     recall_save_cookie_multi_key );

/* Usage */
char *recall_load_keystrokes_multi_close(
     char RECALL_DELIMITER,
     char WIDGET_MULTI_LEFT_RIGHT_DELIMITER );

/* Process */
sprintf( static close[],
     "','%c','%c','%c')",
     RECALL_DELIMITER,
     WIDGET_MULTI_LEFT_RIGHT_DELIMITER,
     RECALL_DELIMITER
          /* multi_select_remember_delimiter */ );
```

FORM

```
/* Usage */
char *form_multi_select_all_javascript(
    LIST *widget_container_list );

/* Process */
char javascript[];
char *ptr = javascript;

for WIDGET_CONTAINER *widget_container in
    widget_container_list
{
    if (widget_container->widget_type !=
        multi_drop_down )
    {
        continue;
    }

    if ( ptr != javascript ) ptr += sprintf( ptr, " && " );

    ptr += sprintf( ptr,
        "post_change_multi_select_all('%s')",
        widget_container->widget_name );
}

/* Usage */
char *form_title_tag(
    char *title );

/* Process */

/* Usage */
char *form_tag(
    char *form_name,
    char *action_string,
    char *target_frame );

/* Process */

/* Usage */
char *form_close_tag(
    void );

/* Process */

/* Usage */
char *form_next_reference_number(
    int *form_current_reference_number );

/* Process */

/* Usage */
char *form_cookie_key(
    char *form_name,
    char *folder_name );

/* Process */

/* Usage */
char *form_cookie_multi_key(
    char *form_name,
    char *folder_name );

/* Process */

/* Usage */
char *form_verify_notepad_widths_javascript(
    char *form_name,
    LIST *widget_container_list );

/* Process */

/* Usage */
char *form_no_display_name(
    const char *
        DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
    char *folder_attribute->attribute_name );

/* Process */

/* Usage */
char *form_ignore_name(
    const char *DICTIONARY_SEPARATE_IGNORE_PREFIX,
    char *relation->name );

/* Process */

/* Usage */
char **form_background_color_array(
    int *background_color_array_length );

/* Process */

/* Usage */
char *form_background_color( void );

/* Process */

/* Usage */
char *form_prefix_widget_name(
    char *prefix,
    char *attribute_name );

/* Process */

/* Usage */
char *form_html(
    char *form_tag(),
    char *widget_container_list_html(),
    char *hidden_html,
    char *form_close_tag() );

/* Process */

/* Usage */
int form_increment_number(
    pid_t process_id );

/* Process */
static int ++increment_number;

return process_id + increment_number;

/* Usage */
char *form_name(
    char *label,
    int form_increment_number );

/* Process *
```

FORM_TABLE_INSERT (1)

```
/* Usage */
FORM_TABLE_INSERT *form_table_insert_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *target_frame,
    int table_insert_rows_number(),
    char *folder->post_change_javascript,
    LIST *folder_attribute_non_primary_name_list,
    LIST *relation_mto_list(),
    LIST *relation_mto1_isa_list(),
    LIST *folder_attribute_append_isa_list(),
    DICTIONARY *drillthru_dictionary,
    DICTIONARY *prompt_dictionary,
    DICTIONARY *query_dictionary,
    DICTIONARY *pair_dictionary,
    LIST *ignore_name_list,
    LIST *prompt_name_list,
    SECURITY_ENTITY *security_entity );

/* Process */
FORM_TABLE_INSERT *form_table_insert_calloc( void );

char *form_table_insert_action_string(
    char *POST_TABLE_INSERT_EXECUTABLE,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *target_frame );

char *form_tag(
    FORM_TABLE_INSERT_FORM_NAME,
    form_table_insert_action_string(),
    target_frame );

FORM_TABLE_INSERT_WIDGET_LIST *
    form_table_insert_widget_list =
        form_table_insert_widget_list_new(
            application_name,
            session_key,
            login_name,
            role_name,
            folder_name /* insert_folder_name */,
            post_change_javascript,
            folder_attribute_non_primary_name_list,
            relation_mto1_list,
            relation_mto1_isa_list,
            folder_attribute_append_isa_list,
            drillthru_dictionary,
            prompt_dictionary,
            ignore_name_list,
            prompt_name_list,
            security_entity );

LIST *widget_container_heading_label_list(
    form_table_insert_widget_list->
        widget_container_list );

LIST *widget_container_heading_list(
    widget_container_heading_label_list() );

char *drillthru_hidden_html =
    dictionary_separate_hidden_html(
        DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
        drillthru_dictionary );

char *prompt_hidden_html =
    widget_dictionary_hidden_html(
        prompt_dictionary );

char *query_hidden_html =
    dictionary_separate_hidden_html(
        DICTIONARY_SEPARATE_QUERY_PREFIX,
        query_dictionary );

char *pair_one2m_hidden_html =
    dictionary_separate_hidden_html(
        DICTIONARY_SEPARATE_PAIR_PREFIX,
        pair_dictionary );

LIST *form_table_insert_button_list( void );

char *button_list_html(
    form_table_insert_button_list() );

char *form_table_insert_html(
    FORM_TABLE_INSERT_DEFAULT_ROWS_NUMBER,
    session_key,
    login_name,
    role_name,
    table_insert_rows_number,
    form_tag(),
    form_table_insert_widget_list->
        form_table_insert_relation_list,
    widget_table_open_tag(
        1 /* border_boolean */,
        0 /* cell_spacing */,
        0 /* cell_padding */ ),
    widget_container_heading_list(),
    form_table_insert_widget_list->
        widget_container_list,
    widget_table_close_tag(),
    drillthru_hidden_html,
    prompt_hidden_html,
    query_hidden_html,
    pair_one2m_hidden_html,
    button_list_html,
    form_close_tag() );

/* Attributes */
char *action_string;
char *form_tag;
FORM_TABLE_INSERT_WIDGET_LIST *
    form_table_insert_widget_list;
LIST *widget_container_heading_label_list;
LIST *widget_container_heading_list;
char *drillthru_hidden_html;
char *prompt_hidden_html;
char *query_hidden_html;
char *pair_one2m_hidden_html;
LIST *button_list;
char *button_list_html;
char *html;
```



FORM_TABLE_INSERT (2)

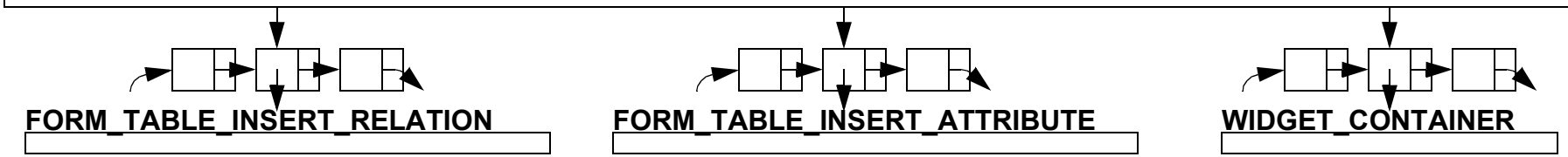
```
/* Usage */
char *form_table_insert_html(
    int FORM_TABLE_INSERT_DEFAULT_ROWS_NUMBER,
    char *session_key,
    char *login_name,
    char *role_name,
    int table_insert_rows_number,
    char *form_tag(),
    LIST *form_table_insert_widget_list->
        form_table_insert_relation_list,
    char *widget_table_open_tag(),
    LIST *widget_container_heading_list(),
    LIST *form_table_insert_widget_list->
        widget_container_list,
    char *widget_table_close_tag(),
    char *drillthru_hidden_html,
    char *prompt_hidden_html,
    char *query_hidden_html,
    char *pair_one2m_hidden_html,
    char *button_list_html,
    char *form_close_tag() );

/* Process */
char html[];
char *ptr = html;

LIST *form_table_insert_relation_ajax_client_list(
    form_table_insert_relation_list );

if ( list_length(
        form_table_insert_relation_ajax_client_list() ) )
{
    char *ajax_client_list_javascript(
        session_key,
        login_name,
        role_name,
        form_table_insert_relation_ajax_client_list() );

    ptr += sprintf( ptr,
        "%s\n",
        ajax_client_list_javascript() );

    free( ajax_client_list_javascript() );
}

ptr += sprintf( ptr,
    "%s\n%s\n",
    form_tag,
    widget_table_open_tag );

char *widget_container_list_html(
    (char *)0 /* state */,
    -1 /* row_number */,
    (char *)0 /* background_color */,
    widget_container_heading_list );

ptr += sprintf( ptr,
    "%s\n",
    widget_container_list_html() );

free( widget_container_list_html() );

if ( table_insert_rows_number <= 0 )
{
    table_insert_rows_number =
        FORM_TABLE_INSERT_DEFAULT_ROWS_NUMBER;
}

for row_number = 1 to table_insert_rows_number
{
    char *widget_container_list_html(
        APPASERVER_INSERT_STATE,
        row_number,
        form_background_color(),
        widget_container_list );

    ptr += sprintf( ptr,
        "%s\n",
        widget_container_list_html() );

    free( widget_container_list_html() );
}

ptr += sprintf( ptr,
    "%s\n",
    widget_table_close_tag );

ptr += sprintf(
    ptr,
    "%s\n%s\n%s\n%s\n%s\n%s",
    drillthru_hidden_html,
    prompt_hidden_html,
    query_hidden_html,
    pair_one2m_hidden_html,
    button_list_html,
    form_close_tag );

/* Usage */
char *form_table_insert_hidden_html(
    DICTIONARY *dictionary );

/* Process */
LIST *widget_container_dictionary_hidden_list(
    dictionary );

char *widget_container_list_hidden_html(
    -1 /* row_number */,
    widget_container_dictionary_hidden_list() );

/* Driver */
printf( "%s\n", form_table_insert->html );
```

FORM_TABLE_INSERT_WIDGET_LIST (1)

```
/* Usage */
FORM_TABLE_INSERT_WIDGET_LIST *
    form_table_insert_widget_list_new(
        char *application_name,
        char *session_key,
        char *login_name,
        char *role_name,
        char *insert_folder_name,
        char *post_change_javascript,
        LIST *folder_attribute_non_primary_name_list,
        LIST *relation_mto1_list(),
        LIST *relation_mto1_isa_list(),
        LIST *folder_append_isa_attribute_list(),
        DICTIONARY *drillthru_dictionary,
        DICTIONARY *prompt_dictionary,
        LIST *ignore_name_list,
        LIST *prompt_name_list,
        SECURITY_ENTITY *security_entity );

/* Process */
FORM_TABLE_INSERT_WIDGET_LIST *
    form_table_insert_widget_list_calloc(
        void );

LIST *widget_container_list = list_new();

list_set(
    widget_container_list,
    widget_container_new(
        table_row, (char *)0 ) );

DICTIONARY *process_command_line_dictionary(
    drillthru_dictionary,
    prompt_dictionary );

for FOLDER_ATTRIBUTE *folder_attribute in
    folder_attribute_append_isa_list
{
    if ( list_string_exists(
            folder_attribute->attribute_name,
            ignore_name_list ) )
                continue;

    if ( list_string_exists(
            folder_attribute->attribute_name,
            prompt_name_list ) )
                continue;

    if ( folder_attribute->omit_insert ) continue;

    if ( ( FORM_TABLE_INSERT_RELATION *
            form_table_insert_relation =
                form_table_insert_relation_new(
                    application_name,
                    session_key,
                    login_name,
                    role_name,
                    insert_folder_name,
                    post_change_javascript,
                    folder_attribute_non_primary_name_list,
                    relation_mto1_list
                        /* many_folder_relation_mto1_list */,
                    relation_mto1_isa_list,
                    process_command_line_dictionary(),
                    security_entity,
                    form_table_insert_relation_list,
                    folder_attribute->attribute_name,
                    folder_attribute->primary_key_index  ) ) )
    {
        if ( !form_table_insert_relation_list )
            form_table_insert_relation_list =
                list_new();

        list_set(
            form_table_insert_relation_list,
            form_table_insert_relation );

        list_set_list(
            widget_container_list,
            form_table_insert_relation->
                widget_container_list );

        continue;
    }
```

FORM_TABLE_INSERT_WIDGET_LIST (2)

```
/* Process (continued) */
    if ( ( FORM_TABLE_INSERT_ATTRIBUTE *
           form_table_insert_attribute =
              form_table_insert_attribute_new(
                 application_name,
                 login_name,
                 post_change_javascript,
                 folder_attribute->attribute_name,
                 folder_attribute->prompt,
                 folder_attribute->
                    attribute->
                    datatype_name,
                 folder_attribute->
                    attribute->
                    width,
                 folder_attribute->
                    attribute->
                    hint_message,
                 form_table_insert_relation_list ) ) )
{
    if ( !form_table_insert_attribute_list )
        form_table_insert_attribute_list =
            list_new();

    list_set(
        form_table_insert_attribute_list,
        form_table_insert_attribute );

    list_set_list(
        widget_container_list,
            form_table_insert_attribute->
                widget_container_list );
    }
}

list_set_list(
    widget_container_list,
    widget_container_dictionary_hidden_list(
        prompt_dictionary ) );

/* Attributes */
LIST *widget_container_list;
DICTIONARY *process_command_line_dictionary;
LIST *form_table_insert_relation_list;
LIST *form_table_insert_attribute_list;
```



FORM_TABLE_INSERT_RELATION | **FORM_TABLE_INSERT_ATTRIBUTE** | **WIDGET_CONTAINER**

FORM_TABLE_INSERT_AUTOMATIC (1)

```
/* Usage */
FORM_TABLE_INSERT_AUTOMATIC *
    form_table_insert_automatic_new(
        char *application_name,
        char *session_key,
        char *login_name,
        char *role_name,
        char *folder_name,
        char *target_frame,
        char *post_change_javascript,
        LIST *many_folder_relation_mto1_list,
        LIST *relation_mto1_isa_list(),
        LIST *folder_attribute_append_isa_list(),
        DICTIONARY *drillthru_dictionary,
        DICTIONARY *prompt_dictionary,
        DICTIONARY *pair_dictionary,
        LIST *prompt_name_list,
        SECURITY_ENTITY *security_entity,
        RELATION_MTO1 *
            relation_mto1_automatic_preselection,
        LIST *query_dictionary_delimited_list );

/* Process */
FORM_TABLE_INSERT_AUTOMATIC *
    form_table_insert_automatic_calloc(
        void );

char *form_table_insert_action_string(
    POST_TABLE_INSERT_EXECUTABLE,
    application_name,
    session_key,
    login_name,
    role_name,
    folder_name,
    target_frame );

char *form_tag(
    FORM_TABLE_INSERT_FORM_NAME,
    form_table_insert_action_string(),
    FRAMESET_TABLE_FRAME );

LIST *relation_mto1_to_one_list =
    relation_mto1_list(
        relation_mto1_automatic_preselection->
            one_folder_name
                /* many_folder_name */,
        relation_mto1_automatic_preselection->
            one_folder->
                folder_attribute_primary_key_list
                    /* many_folder_primary_key_list */ );

DICTIONARY *process_command_line_dictionary(
    drillthru_dictionary,
    prompt_dictionary );

QUERY_DROP_DOWN *query_drop_down_new(
    application_name,
    session_key,
    login_name,
    role_name,
    APPASERVER_INSERT_STATE,
    folder_name /* many_folder_name */,
    relation_mto1_automatic_preselection->
        one_folder_name,
    (char *)0 /* related_column */,
    relation_mto1_automatic_preselection->
        one_folder->
        folder_attribute_list,
    relation_mto1_automatic_preselection->
        one_folder->
        populate_drop_down_process_name,
    relation_mto1_to_one_list,
    process_command_line_dictionary()
        /* drop_down_dictionary */,
    security_entity,
    (LIST *)0 /* common_name_list */ );

FORM_TABLE_INSERT_AUTOMATIC_WIDGET_LIST *
    form_table_insert_automatic_widget_list =
        form_table_insert_automatic_widget_list_new(
            application_name,
            session_key,
            login_name,
            role_name,
            folder_name /* insert_folder_name */,
            post_change_javascript,
            many_folder_relation_mto1_list,
            relation_mto1_isa_list,
            folder_attribute_append_isa_list,
            process_command_line_dictionary(),
            pair_dictionary,
            prompt_name_list,
            security_entity,
            relation_mto1_automatic_preselection->
                relation_name );

LIST *widget_container_heading_label_list(
    form_table_insert_automatic_widget_list->
        widget_container_list );

list_set_first(
    widget_container_heading_label_list(),
    "key" );

LIST *widget_container_heading_list(
    widget_container_heading_label_list() );

char *prompt_hidden_html =
    widget_dictionary_hidden_html(
        prompt_dictionary );

DICTIONARY *dictionary_prefix(
    pair_dictionary,
    DICTIONARY_SEPARATE_PAIR_PREFIX );

char *pair_one2m_hidden_html =
    widget_dictionary_hidden_html(
        dictionary_prefix() );

LIST *form_table_insert_button_list();

char *button_list_html(
    form_table_insert_button_list() );

char *form_table_insert_automatic_html(
    relation_mto1_automatic_preselection->
        one_folder->
        no_initial_capital,
    query_dictionary_delimited_list,
    form_tag(),
    widget_table_open_tag(
        1 /* border_boolean */,
        0 /* cell_spacing,
        0 /* cell_padding ),
    widget_container_heading_list(),
    query_drop_down_new()->delimited_list,
    form_table_insert_automatic_widget_list
        /* in/out */,
    widget_table_close_tag(),
    prompt_hidden_html,
    pair_one2m_hidden_html,
    button_list_html(),
    form_close_tag() );
```

FORM_TABLE_INSERT_AUTOMATIC (2)

```
/* Usage */
char *form_table_insert_automatic_html(
    boolean no_initial_capital,
    LIST *query_dictionary_delimited_list,
    char *form_tag,
    char *widget_table_open_tag(),
    LIST *widget_container_heading_list(),
    LIST *query_drop_down_delimited_list,
    FORM_TABLE_INSERT_AUTOMATIC_WIDGET_LIST *
        form_table_insert_automatic_widget_list
            /* in/out */,
    char *widget_table_close_tag(),
    char *prompt_hidden_html,
    char *pair_one2m_hidden_html,
    char *button_list_html(),
    char *form_close_tag() );

/* Process */
char html[];
char *ptr = html;

ptr += sprintf( ptr,
    "%s\n%s\n",
    form_tag,
    widget_table_open_tag );

char *widget_container_list_html(
    (char *)0 /* state */,
    -1 /* row_number */,
    (char *)0 /* background_color */,
    widget_container_heading_list );

ptr += sprintf( ptr,
    "%s\n",
    widget_container_list_html() );

free( widget_container_list_html() );

char *form_table_insert_automatic_widget_html(
    no_initial_capital,
    query_dictionary_delimited_list,
    query_drop_down_delimited_list,
    form_table_insert_automatic_widget_list /* in/out */ );

char *form_table_insert_automatic_hidden_html(
    query_dictionary_delimited_list,
    query_drop_down_delimited_list,
    form_table_insert_automatic_widget_list /* in/out */ );

ptr += sprintf( ptr,
    "%s\n%s\n%s\n%s\n%s\n%s\n%s",
    form_table_insert_automatic_widget_html(),
    widget_table_close_tag,
    prompt_hidden_html,
    pair_one2m_hidden_html,
    form_table_insert_automatic_hidden_html(),
    button_list_html,
    form_close_tag );

free( form_table_insert_automatic_widget_html() );

free( form_table_insert_automatic_hidden_html() );

/* Usage */
char *form_table_insert_automatic_widget_html(
    boolean no_initial_capital,
    LIST *query_dictionary_delimited_list,
    LIST *query_drop_down_delimited_list,
    FORM_TABLE_INSERT_AUTOMATIC_WIDGET_LIST *
        form_table_insert_automatic_widget_list
        /* in/out */ );

/* Process */
char html[];
char *ptr = html;
int row_number = 0;

for char *delimited_string in
    query_drop_down_delimited_list
{

    char *widget_drop_down_option_value_string(
        WIDGET_DROP_DOWN_LABEL_DELIMITER /* '|' */,
        WIDGET_DROP_DOWN_EXTRA_DELIMITER /* '[' */,
        delimited_string );

    if ( list_exists_string(
            widget_drop_down_option_value_string(),
            query_dictionary_delimited_list ) )
    {
        continue;
    }

    char *widget_drop_down_option_display_string(
        WIDGET_DROP_DOWN_LABEL_DELIMITER /* '|' */,
        WIDGET_DROP_DOWN_DASH_DELIMITER,
        no_initial_capital,
        sql_delimiter_string( SQL_DELIMITER ),
        delimited_string );

    form_table_insert_automatic_widget_list->
        non_edit_text_widget_container->
        non_edit_text->
        widget_name =
            widget_drop_down_option_display_string() );

    char *widget_container_list_html(
        APPASERVER_INSERT_STATE,
        ++row_number,
        form_background_color(),
        form_table_insert_automatic_widget_list->
            widget_container_list );

    ptr += sprintf( ptr,
        "%s\n",
        widget_container_list_html() );

    free( widget_drop_down_option_display_string() );

    free( widget_container_list_html() );
}
```

FORM_TABLE_INSERT_AUTOMATIC (3)

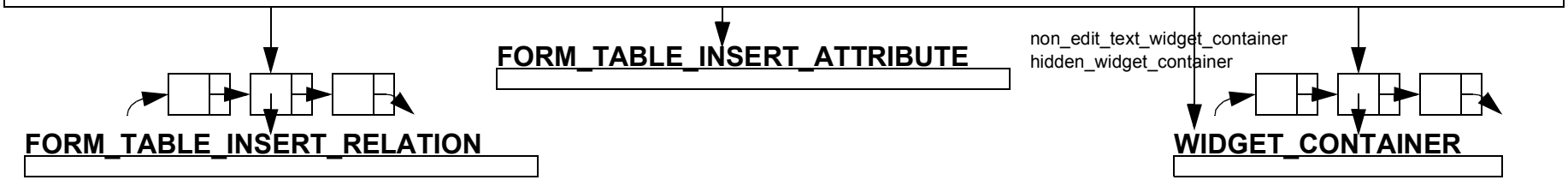
```
/* Usage */
char *form_table_insert_automatic_hidden_html(
    LIST *query_dictionary_delimited_list,
    LIST *query_drop_down_delimited_list,
    FORM_TABLE_INSERT_AUTOMATIC_WIDGET_LIST *
        form_table_insert_automatic_widget_list
        /* in/out */ );

/* Process */
char html[];
char *ptr = html;
int row_number = 0;

for char *delimited_string in
    query_drop_down_delimited_list
{
        char *widget_drop_down_option_value_string(
            WIDGET_DROP_DOWN_LABEL_DELIMITER
                /* probably '|' */,
            WIDGET_DROP_DOWN_EXTRA_DELIMITER
                /* probably '[' */,
            delimited_string ) );

if ( list_exists_string(
        widget_drop_down_option_value_string(),
        query_dictionary_delimited_list ) )
{
    continue;
}

form_table_insert_automatic_widget_list->
    hidden_widget_container->
    hidden->
    value_string =
        char *widget_drop_down_option_value_string() );

char *widget_container_list_hidden_html(
    ++row_number,
    form_table_insert_automatic_widget_list->
        /* Only one is hidden */
        widget_container_list );

ptr += sprintf( ptr,
    "%s",
    /* Has trailing CR */
    widget_container_list_hidden_html() );
    free( widget_container_list_hidden_html() );
}

/* Driver */
printf( "%s\n", form_table_insert_automatic->html );

/* Attributes */
char *form_table_insert_action_string;
char *form_tag;
DICTIONARY *process_command_line_dictionary;
QUERY_DROP_DOWN *query_drop_down;
FORM_TABLE_INSERT_AUTOMATIC_WIDGET_LIST *
    form_table_insert_automatic_widget_list;
LIST *widget_container_heading_label_list;
LIST *widget_container_heading_list;
char *prompt_hidden_html;
char *pair_one2m_hidden_html;
LIST *form_table_insert_button_list;
char *button_list_html;
char *html;
```

FORM_TABLE_INSERT_AUTOMATIC_WIDGET_LIST (1)

```
/* Usage */
FORM_TABLE_INSERT_AUTOMATIC_WIDGET_LIST *
    form_table_insert_automatic_widget_list_new(
        char *application_name,
        char *session_key,
        char *login_name,
        char *role_name,
        char *insert_folder_name,
        char *post_change_javascript,
        LIST *many_folder_relation_mto1_list,
        LIST *relation_mto1_isa_list(),
        LIST *folder_attribute_append_isa_list(),
        DICTIONARY *process_command_line_dictionary(),
        LIST *prompt_name_list,
        SECURITY_ENTITY *security_entity,
        char *
            relation_mto1_automatic_preselection_name );

/* Process */
FORM_TABLE_INSERT_AUTOMATIC_WIDGET_LIST *
    form_table_insert_automatic_widget_list_calloc(
        void );

LIST *widget_container_list = list_new();

list_set(
    widget_container_list,
    widget_container_new(
        table_row, (char *)0 ) );

WIDGET_CONTAINER *table_data_widget_container =
    widget_container_new(
        table_data,
        (char *)0 );

table_data_widget_container->
    table_data->
        width_em = 35;

list_set(
    widget_container_list,
    table_data_widget_container );

WIDGET_CONTAINER *
    non_edit_text_widget_container =
        widget_container_new(
            non_edit_text,
            (char *)0 /* widget_name */ );

list_set(
    widget_container_list,
    non_edit_text_widget_container );

WIDGET_CONTAINER *
    hidden_widget_container =
        widget_container_new(
            hidden,
            relation_mto1_automatic_preselection_name
                /* widget_name */ );

list_set(
    widget_container_list,
    hidden_widget_container );

for FOLDER_ATTRIBUTE *folder_attribute in
    folder_attribute_append_isa_list
{
    if ( list_string_exists(
            folder_attribute->attribute_name,
            prompt_name_list ) )
    {
        continue;
    }

if ( folder_attribute->primary_key_index ) continue;

if ( folder_attribute->omit_insert ) continue;

if ( ( FORM_TABLE_INSERT_RELATION *
        form_table_insert_relation =
            form_table_insert_relation_new(
                application_name,
                session_key,
                login_name,
                role_name,
                insert_folder_name,
                post_change_javascript,
                (LIST *)0 /* non_primary_name_list */,
                many_folder_relation_mto1_list,
                relation_mto1_isa_list,
                process_command_line_dictionary,
                security_entity,
                form_table_insert_relation_list,
                folder_attribute->attribute_name,
                folder_attribute->primary_key_index  ) ) )
{
    if ( !form_table_insert_relation_list )
        form_table_insert_relation_list =
            list_new();

    list_set(
        form_table_insert_relation_list,
        form_table_insert_relation );

    list_set_list(
        widget_container_list,
        form_table_insert_relation->
            widget_container_list );

    continue;
}
```

FORM_TABLE_INSERT_AUTOMATIC_WIDGET_LIST (2)

```
/* Process (continued) */
    if ( ( FORM_TABLE_INSERT_ATTRIBUTE *
           form_table_insert_attribute =
              form_table_insert_attribute_new(
                 application_name,
                 login_name,
                 post_change_javascript,
                 folder_attribute->attribute_name,
                 folder_attribute->prompt,
                 folder_attribute->
                    attribute->
                       datatype_name,
                 folder_attribute->
                    attribute->
                    width,
                 form_table_insert_relation_list ) ) )
{
    list_set_list(
       widget_container_list,
       form_table_insert_attribute->
          widget_container_list );
}
}

/* Attributes */
WIDGET_CONTAINER *non_edit_text_widget_container;
WIDGET_CONTAINER *hidden_widget_container;
LIST *form_table_insert_relation_list;
FORM_TABLE_INSERT_ATTRIBUTE *
   form_table_insert_attribute;
LIST *widget_container_list;
```

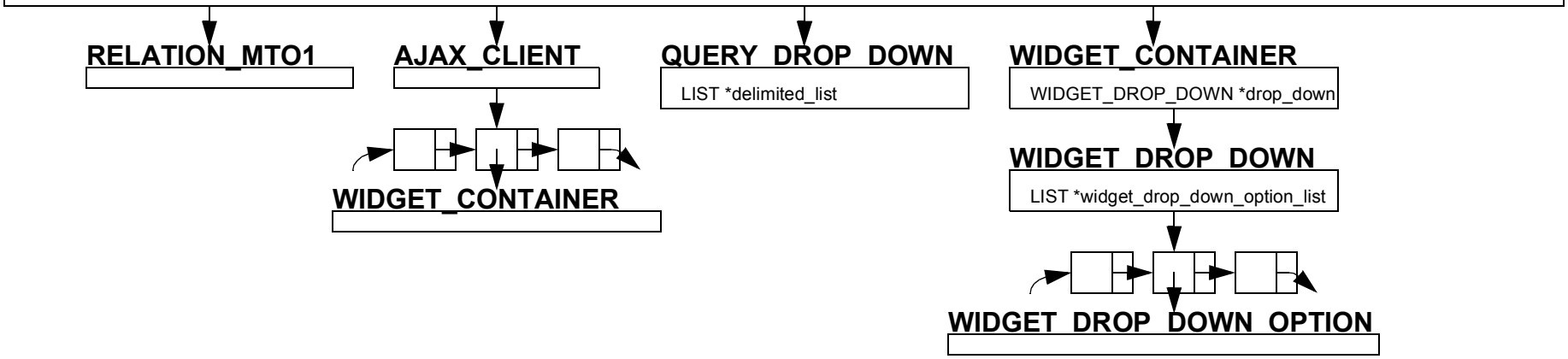


FORM_TABLE_INSERT_RELATION (1)

```
/* Usage */
FORM_TABLE_INSERT_RELATION *
    form_table_insert_relation_new(
        char *application_name,
        char *session_key,
        char *login_name,
        char *role_name,
        char *insert_folder_name,
        char *post_change_javascript,
        LIST *folder_attribute_non_primary_name_list,
        LIST *many_folder_relation_mto1_list,
        LIST *relation_mto1_isa_list,
        DICTIONARY *process_command_line_dictionary,
        SECURITY_ENTITY *security_entity,
        LIST *form_table_insert_relation_list,
        char *folder_attribute->attribute_name,
        int folder_attibute->primary_key_index );

/* Process */
boolean form_table_insert_relation_attribute_exists(
    attribute_name,
    form_table_insert_relation_list );

if ( form_table_insert_relation_attribute_exists() )
        return NULL;

FORM_TABLE_INSERT_RELATION *
    form_table_insert_relation_calloc(
        void );

RELATION_MTO1 *relation_mto1 =
    relation_mto1_consumes(
        attribute_name,
        many_folder_relation_mto1_list );

if ( !relation_mto1 )
{
    RELATION_MTO1 *relation_mto1 =
        relation_mto1_isa_consumes(
            attribute_name,
            relation_mto1_isa_list );

    if ( !relation_mto1 ) return NULL;
}

if (    primary_key_index
&&  relation_mto1->
            relation->
            relation_type_isa )
{
    return NULL;
}

LIST *widget_container_list = list_new();

list_set(
    widget_container_list,
    widget_container_new(
        table_data, (char *)0 ) );

if ( relation_mto1->
        relation->
        ajax_fill_drop_down )
{
    AJAX_CLIENT *ajax_client =
        form_table_insert_relation_ajax_client(
            role_name,
            relation_mto1 );
}

if ( ajax_client )
{
    list_set_list(
        widget_container_list,
        ajax_client->widget_container_list );

    return this;
}

list_set(
    widget_container_list,
    ( drop_down_widget_container =
        widget_container_new(
            drop_down,
            relation_mto1->relation_name
                /* widget_name */ ) ) );

drop_down_widget_container->
    drop_down->
    post_change_javascript =
        post_change_javascript;

drop_down_widget_container->heading_boolean = 1;

LIST *relation_mto1_to_one_list =
    relation_mto1_list(
        relation_mto1->
            one_folder_name
                /* many_folder_name */,
        relation_mto1->
            one_folder->
            folder_attribute_primary_key_list
                /* many_folder_primary_key_list */ );
```

FORM_TABLE_INSERT_RELATION (2)

```
/* Process (continued) */
LIST *relation_mto1_common_name_list(
    folder_attribute_non_primary_name_list,
    relation_mto1->
        relation->
        copy_common_columns,
    relation_mto1->
        one_folder->
        folder_attribute_name_list );

QUERY_DROP_DOWN *query_drop_down_new(
    application_name,
    session_key,
    login_name,
    role_name,
    APPASERVER_INSERT_STATE,
    insert_folder_name
        /* many_folder_name */,
    relation_mto1->
        one_folder_name,
    relation_mto1->relation->related_attribute_name
        /* related_column */,
    relation_mto1->
        one_folder->
        folder_attribute_list,
    relation_mto1->
        one_folder->
        populate_drop_down_process_name,
    relation_mto1_to_one_list,
    process_command_line_dictionary
        /* drop_down_dictionary */,
    security_entity,
    relation_mto1_common_name_list() );

drop_down_widget_container->
    drop_down->
    widget_drop_down_option_list =
        LIST *widget_drop_down_option_list(
            ATTRIBUTE_MULTI_KEY_DELIMITER,
            WIDGET_DROP_DOWN_LABEL_DELIMITER,
            WIDGET_DROP_DOWN_EXTRA_DELIMITER,
            WIDGET_DROP_DOWN_DASH_DELIMITER,
            query_drop_down_new()->delimited_list,
            relation_mto1->
                one_folder->
                no_initial_capital );

drop_down_widget_container->
    drop_down->
    top_select_boolean = 1;

/* Attributes */
RELATION_MTO1 *relation_mto1;
AJAX_CLIENT *ajax_client;
LIST *relation_mto1_common_name_list;
QUERY_DROP_DOWN *query_drop_down;
WIDGET_CONTAINER *drop_down_widget_container;
LIST *widget_container_list;
```



FORM_TABLE_INSERT_RELATION (3)

```
/* Usage */
boolean form_table_insert_relation_attribute_exists(
      char *attribute_name,
      LIST *form_table_insert_relation_list );

/* Process */
for FORM_TABLE_INSERT_RELATION *
    form_table_insert_relation in
        form_table_insert_relation_list
{
    boolean list_string_exists(
        attribute_name,
        form_table_insert_relation->
            relation_mto1->
            relation_foreign_key_list );

    if ( list_string_exists() ) return 1;
}
return 0;
```

```
/* Usage */
AJAX_CLIENT *form_table_insert_relation_ajax_client(
    char *role_name,
    RELATION_MTO1 *relation_mto1 );

/* Process */
LIST *relation_mto1_to_one_fetch_list(
    relation_mto1->one_folder_name,
    relation_mto1->
        one_folder->
        folder_attribute_primary_key_list );

AJAX_CLIENT *
    /* Returns null if not participating */
    ajax_client_relation_mto1_new(
        relation_mto1,
        relation_mto1_to_one_fetch_list(),
        1 /* top_select_boolean */ );

/* Usage */
```

```
LIST *form_table_insert_relation_ajax_client_list(
    LIST *form_table_insert_relation_list );

/* Process */
LIST *ajax_client_list = {0};

for FORM_TABLE_INSERT_RELATION *
    form_table_insert_relation in
        form_table_insert_relation_list
{
    if ( form_table_insert_relation->ajax_client )
    {
        if ( !ajax_client_list ) ajax_client_list = list_new();

        list_set(
            ajax_client_list,
            form_table_insert_relation->ajax_client );
    }
}
```

FORM_TABLE_INSERT_ATTRIBUTE (1)

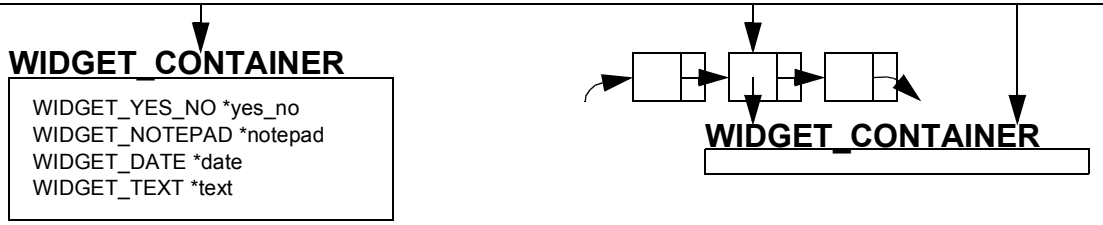
```
/* Usage */
FORM_TABLE_INSERT_ATTRIBUTE *
    form_table_insert_attribute_new(
        char *application_name,
        char *login_name,
        char *post_change_javascript,
        char *attribute_name,
        char *folder_attribute_prompt,
        char *datatype_name,
        int attribute_width,
        LIST *form_table_insert_relation_list );

/* Process */
if ( form_table_insert_relation_attribute_exists(
    attribute_name,
    form_table_insert_relation_list ) )
{
    return NULL;
}

FORM_TABLE_INSERT_ATTRIBUTE *
    form_table_insert_attribute_calloc(
        void );
```

```
LIST *widget_container_list = list_new();

list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );

if ( attribute_is_yes_no( attribute_name ) )
{
    list_set(
        widget_container_list,
        ( widget_container =
            widget_container_new(
                yes_no,
                attribute_name
                    /* widget_name */ ) ) );

    widget_container->
        yes_no->
        post_change_javascript =
            post_change_javascript;
}
```

```
else
if ( attribute_is_notepad(
        ATTRIBUTE_DATATYPE_NOTEPAD,
        datatype_name ) )
{
    list_set(
        widget_container_list,
        ( widget_container =
            widget_container_new(
                notepad,
                attribute_name
                    /* widget_name */ ) ) );

    widget_container->
        notepad->
        attribute_size = attribute_width;

    widget_container->
        notepad->
        post_change_javascript =
            post_change_javascript;
}
```

FORM_TABLE_INSERT_ATTRIBUTE (2)

```
/* Process (continued) */
else
if (    attribute_is_date( datatype_name )
||      attribute_is_date_time( datatype_name )
||      attribute_is_current_date_time( datatype_name ) )
{
    list_set(
        widget_container_list,
        ( widget_container =
            widget_container_new(
                widget_date,
                attribute_name
                    /* widget_name */ ) ) );

    widget_container->
        date->
        datatype_name = datatype_name;

    widget_container->
        date->
        attribute_width_max_length =
            attribute_width;

    widget_container->
        date->
        display_size =
            attribute_width;

    widget_container->
        date->
        application_name = application_name;

    widget_container->
        date->
        login_name = login_name;

    widget_container->
        date->
        post_change_javascript =
            post_change_javascript;
}
else
{
    list_set(
        widget_container_list,
        ( widget_container =
            widget_container_new(
                character,
                attribute_name
                    /* widget_name */ ) ) );

    widget_container->
        text->
        datatype_name = datatype_name;

    int widget_text_display_size(
        WIDGET_TEXT_DEFAULT_MAX_LENGTH,
        attribute_width );

    widget_container->
        text->
        widget_text_display_size =
            widget_text_display_size();

    widget_container->
        text->
        attribute_width_max_length = attribute_width;

    widget_container->
        text->
        post_change_javascript =
            post_change_javascript;
}

widget_container->heading_boolean = 1;

/* Attributes */
WIDGET_CONTAINER *widget_container;
LIST *widget_container_list;
```



FORM_PROMPT_ATTRIBUTE (1)

```
/* Usage */
FORM_PROMPT_ATTRIBUTE *
    form_prompt_attribute_new(
        char *application_name,
        char *login_name,
        char *attribute_name,
        char *folder_attribute_prompt,
        char *datatype_name,
        int attribute_width,
        char *hint_message );

/* Process */
FORM_PROMPT_ATTRIBUTE *
    form_prompt_attribute_calloc(
        void );

LIST *widget_container_list = list_new();

list_set(
    widget_container_list,
    widget_container_new(
        table_data, (char *)0 ) );

list_set(
    widget_container_list,
    widget_container_new(
        non_edit_text,
        folder_attribute_prompt ) );

list_set(
    widget_container_list(
    widget_container_new(
        table_data, (char *)0 ) );

list_set(
    widget_container_list(
    widget_container_new(
        table_open, (char *)0 ) );

list_set(
    widget_container_list(
    widget_container_new(
        table_row, (char *)0 ) );

list_set(
    widget_container_list(
    widget_container_new(
        table_data, (char *)0 ) );

char *form_prefix_widget_name(
    QUERY_RELATION_OPERATOR_PREFIX,
    attribute_name );

list_set(
    widget_container_list,
    ( relation_operator_widget_container =
        widget_container_new(
            drop_down,
            form_prefix_widget_name() ) ) );

LIST *form_prompt_attribute_option_list(
    char *datatype_name );

relation_operator_widget_container->
    drop_down->
    widget_drop_down_option_list =
        form_prompt_attribute_relational_option_list();

relation_operator_widget_container->
    drop_down->
    top_select_boolean = 1;

relation_operator_widget_container->
    drop_down->
    display_size = 1;

relation_operator_widget_container->
    recall_boolean = 1;

list_set(
    widget_container_list(
    widget_container_new(
        table_data, (char *)0 ) );
```

FORM_PROMPT_ATTRIBUTE (2)

```
/* Process (continued) */
if (    attribute_is_number( datatype_name )
||      attribute_is_date( datatype_name )
||      attribute_is_time( datatype_name )
||      attribute_is_date_time( datatype_name )
||      attribute_is_current_date_time( datatype_name ) )
{
    char *form_prefix_widget_name(
        PROMPT_LOOKUP_FROM_PREFIX,
        attribute_name );

    WIDGET_CONTAINER *
    from_widget_container =
        form_prompt_attribute_from_container(
            application_name,
            login_name,
            datatype_name,
            form_prefix_widget_name() );

    list_set(
        widget_container_list,
        from_widget_container );

    list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );

    list_set(
        widget_container_list,
        ( and_widget_container =
            widget_container_new(
                non_edit_text, “and” /* widget_name */ ) ) );

    list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );

    char *form_prefix_widget_name(
        PROMPT_LOOKUP_TO_PREFIX,
        attribute_name );

    WIDGET_CONTAINER *
    to_widget_container =
        form_prompt_attribute_to_container(
            application_name,
            login_name,
            datatype_name,
            attribute_width,
            form_prefix_widget_name() );

    list_set(
        widget_container_list,
        to_widget_container );
}
else
{
    list_set(
        widget_container_list,
        ( from_widget_container =
            widget_container_new(
                character,
                attribute_name ) ) );

    from_widget_container->
        text->
        datatype_name = datatype_name;

    from_widget_container->
        text->
        widget_text_display_size =
            FORM_PROMPT_LOOKUP_FROM_SIZE;

    from_widget_container->
        text->
        attribute_width_max_length =
            FORM_PROMPT_LOOKUP_FROM_MAX_LENGTH;

    from_widget_container->recall_boolean = 1;
}

list_set(
    widget_container_list(
    widget_container_new(
        table_close, (char *)0 ) );

if ( hint_message )
{
    list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );

    list_set(
        widget_container_list,
        ( hint_message_widget_container =
            widget_container_new(
                non_edit_text,
                hint_message
                    /* widget_name */ ) ) );

    hint_message_widget_container->
        non_edit_text->
        no_initial_capital_boolean = 1;
}
```

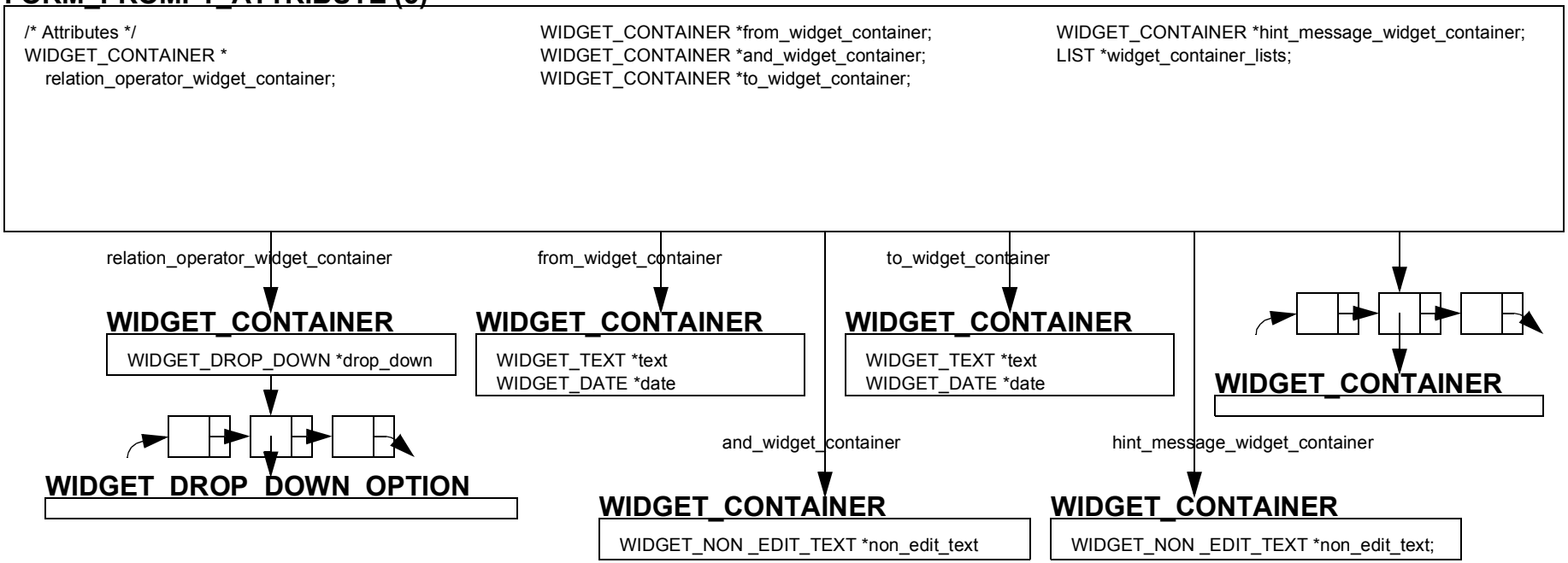
FORM_PROMPT_ATTRIBUTE (3)

```
/* Usage */
WIDGET_CONTAINER *
    form_prompt_attribute_from_container(
        char *application_name,
        char *login_name,
        char *datatype_name,
        char *form_prefix_widget_name() );

/* Process */
if (    attribute_is_date( datatype_name )
||      attribute_is_date_time( datatype_name )
||      attribute_is_current_date_time( datatype_name ) )
{
    from_widget_container =
        widget_container_new(
            widget_date,
            form_prefix_widget_name );

    from_widget_container->
        date->
        application_name =
            application_name;

    from_widget_container->
        date->
        login_name =
            login_name;

    from_widget_container->
        date->
        datatype_name =
            datatype_name;

    from_widget_container->
        date->
        display_size =
            FORM_PROMPT_LOOKUP_FROM_SIZE;

    from_widget_container->
        date->
        attribute_width_max_length =
            FORM_PROMPT_LOOKUP_FROM_MAX_LENGTH;
}
else
{
    WIDGET_CONTAINER *
        from_widget_container =
            widget_container_new(
                character,
                form_prefix_widget_name );

    from_widget_container->
        text->
        datatype_name = datatype_name;

    from_widget_container->
        text->
        widget_text_display_size =
            FORM_PROMPT_LOOKUP_FROM_SIZE;

    from_widget_container->
        text->
        attribute_width_max_length =
            FORM_PROMPT_LOOKUP_FROM_MAX_LENGTH;
}

from_widget_container->recall_boolean = 1;
```

FORM_PROMPT_ATTRIBUTE (4)

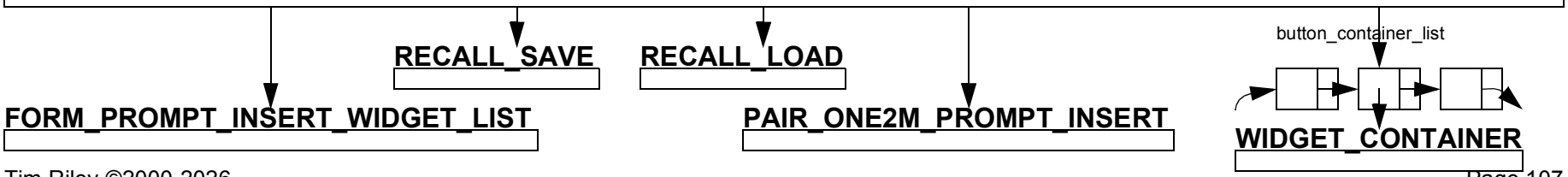
```
/* Usage */
WIDGET_CONTAINER *
    form_prompt_attribute_to_container(
        char *application_name,
        char *login_name,
        char *datatype_name,
        int attribute_width,
        char *form_prefix_widget_name() );

/* Process */
if (    attribute_is_date( datatype_name )
||      attribute_is_date_time( datatype )
||      attribute_is_current_date_time( datatype_name )
{
    to_widget_container =
        widget_container_new(
            widget_date,
            form_prefix_widget_name );

    to_widget_container->
        date->
        application_name =
            application_name;

    to_widget_container->
        date->
        login_name =
            login_name;

to_widget_container->
    date->
    datatype_name =
        datatype_name;

to_widget_container->
    date->
    display_size =
        FORM_PROMPT_LOOKUP_TO_SIZE;

if (    attribute_is_date_time(
            datatype_name )
||      attribute_is_current_date_time(
            datatype_name ) )
{
    to_widget_container->
        date->
        attribute_width_max_length =
            WIDGET_DATE_TIME_DISPLAY_SIZE;
}
else
{
    to_widget_container->
        date->
        attribute_width_max_length =
            WIDGET_DATE_DISPLAY_SIZE;
    }
}
else
{
    WIDGET_CONTAINER *
        to_widget_container =
            widget_container_new(
                character,
                form_prefix_widget_name );

    to_widget_container->
        text->
        datatype_name = datatype_name;

    to_widget_container->
        text->
        widget_text_display_size =
            FORM_PROMPT_LOOKUP_TO_SIZE;

    to_widget_container->
        text->
        attribute_width_max_length =
            attribute_width;
}

to_widget_container->recall_boolean = 1;
```

FORM_PROMPT_ATTRIBUTE (5)



FORM_PROMPT_INSERT (1)

```
/* Usage */
FORM_PROMPT_INSERT *form_prompt_insert_new(
    const char *PROMPT_INSERT_LOOKUP_CHECKBOX,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *post_change_javascript,
    LIST *folder_attribute_non_primary_name_list,
    LIST *relation_mto1_list(),
    LIST *relation_mto1_isa_list(),
    LIST *folder_attribute_append_isa_list(),
    DICTIONARY *drillthru_dictionary,
    SECURITY_ENTITY *security_entity,
    boolean role_folder_lookup_boolean,
    LIST *relation_one2m_pair_list(),
    char *post_choose_folder_action_string );

/* Process */
FORM_PROMPT_INSERT *form_prompt_insert_calloc(
    void );

char *form_prompt_insert_action_string(
    const char *POST_PROMPT_INSERT_EXECUTABLE,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name );

char *form_tag(
    FORM_PROMPT_INSERT_NAME,
    form_prompt_insert_action_string(),
    FRAMESET_TABLE_FRAME );

int relation_one2m_pair_list_length =
    list_length( relation_one2m_pair_list );

boolean form_prompt_insert_ignore_boolean(
    int relation_one2m_pair_list_length );

FORM_PROMPT_INSERT_WIDGET_LIST *
    form_prompt_insert_widget_list =
        form_prompt_insert_widget_list_new(
            PROMPT_INSERT_LOOKUP_CHECKBOX,
            application_name,
            session_key,
            login_name,
            role_name,
            folder_name,
            post_change_javascript,
            folder_attribute_non_primary_name_list,
            relation_mto1_list,
            relation_mto1_isa_list,
            drillthru_dictionary,
            security_entity,
            folder_attribute_append_isa_list,
            role_folder_lookup_boolean,
            form_prompt_insert_ignore_boolean() );

char *form_multi_select_all_javascript(
    form_prompt_insert_widget_list->
        widget_container_list );

char *form_verify_notepad_widths_javascript(
    FORM_PROMPT_INSERT_NAME,
    form_prompt_insert_widget_list->
        widget_container_list );

RECALL_SAVE *recall_save =
    recall_save_new(
        APPASERVER_INSERT_STATE,
        FORM_PROMPT_INSERT_NAME,
        folder_name,
        form_prompt_insert_widget_list->
            widget_container_list,
        application_ssl_support_boolean(
            application_name ) );

RECALL_LOAD *recall_load =
    recall_load_new(
        APPASERVER_INSERT_STATE,
        FORM_PROMPT_INSERT_NAME,
        folder_name,
        form_prompt_insert_widget_list->
            widget_container_list );

if ( list_length(
        form_prompt_insert_widget_list->
            form_prompt_insert_relation_list ) )
{
    char *widget_checkbox_submit_javascript(
        FORM_PROMPT_INSERT_NAME,
        recall_save->javascript );

    form_prompt_insert_relation_javascript_save_set(
        form_prompt_insert_widget_list->
            form_prompt_insert_relation_list,
        widget_checkbox_submit_javascript() );
}

if ( relation_one2m_pair_list_length )
{
    PAIR_ONE2M_PROMPT_INSERT *
        pair_one2m_prompt_insert =
            pair_one2m_prompt_insert_new(
                DICTIONARY_SEPARATE_PAIR_PREFIX,
                PAIR_ONE2M_ONE_FOLDER_KEY,
                PAIR_ONEM_UNFULFILLED_LIST_KEY,
                PAIR_ONE2M_FOLDER_DELIMITER,
                folder_name /* one_folder_name */,
                recall_save->javascript,
                relation_one2m_pair_list );

    list_set(
        form_prompt_insert_widget_list->
            widget_container_list,
        pair_one2m_prompt_insert->hidden_widget );
}
```



FORM_PROMPT_INSERT (2)

```
/* Process (continued) */
LIST *form_prompt_insert_button_container_list(
    post_choose_folder_action_string,
    form_multi_select_all_javascript(),
    recall_save->javascript,
    recall_load->javascript,
    form_verify_notepad_widths_javascript(),
    (pair_one2m_prompt_insert)
        ? pair_one2m_prompt_insert->
            pair_one2m_button_list
        : (LIST *)0 );

char *form_prompt_insert_button_list_html(
    LIST *form_prompt_insert_button_container_list() );

char *form_prompt_insert_html(
    session_key,
    login_name,
    role_name,
    folder_name,
    form_prompt_insert_widget_list->
        form_prompt_insert_relation_list,
    form_tag(),
    form_prompt_insert_widget_list->
        widget_container_list,
    form_prompt_insert_button_list_html(),
    (pair_one2m_prompt_insert)
        ? pair_one2m_prompt_insert->
            pair_dictionary
        : (DICTIONARY *)0,
    form_close_tag() );

free( form_prompt_insert_button_list_html() );

/* Attributes */
char *action_string;
char *form_tag;
int relation_one2m_pair_list_length;
boolean ignore_boolean;
LIST *role_folder_insert_list;
FORM_PROMPT_INSERT_WIDGET_LIST *
    form_prompt_insert_widget_list;
char *form_multi_select_all_javascript;
char *form_verify_notepad_widths_javascript;
RECALL_SAVE *recall_save;
RECALL_LOAD *recall_load;
char *widget_checkbox_submit_javascript;
PAIR_ONE2M_PROMPT_INSERT *
    pair_one2m_prompt_insert;
LIST *button_container_list;
char *button_list_html;
char *html;
```

FORM_PROMPT_INSERT (3)

```
/* Usage */
LIST *form_prompt_insert_button_container_list(
    char *post_choose_folder_action_string,
    char *form_multi_select_all_javascript(),
    char *recall->save_javascript,
    char *recall->load_javascript,
    char *form_verify_notepad_widths_javascript(),
    LIST *pair_one2m_button_list );

/* Process */
LIST *button_container_list = list_new();

WIDGET_CONTAINER *widget_container_new(
    table_open, (char *)0 ) );

list_set(
    button_container_list,
    widget_container_new() );

WIDGET_CONTAINER *widget_container_new(
    table_row, (char *)0 ) );

list_set(
    button_container_list,
    widget_container_new() );

WIDGET_CONTAINER *widget_container_new(
    table_data, (char *)0 ) );

list_set(
    button_container_list,
    widget_container_new() );

WIDGET_CONTAINER *widget_container_new(
    button, (char *)0 ) );

list_set(
    button_container_list,
    button_container_new() );

widget_container_new()->widget_button->button =
    button_submit(
        form_multi_select_all_javascript,
        form_verify_notepad_widths_javascript,
        0 /* form_number */ );

for PAIR_ONE2M_BUTTON *pair_one2m_button in
    pair_one2m_button_list
{
    list_set(
        button_container_list,
        pair_one2m_button->widget_container );
}

WIDGET_CONTAINER *widget_container_new(
    button, (char *)0 );

list_set(
    button_container_list,
    widget_container_new() );

widget_container_new()->widget_button->button =
    button_reset(
        (char *)0 /* javascript_replace */,
        0 /* form_number */ );

WIDGET_CONTAINER *widget_container_new(
    button, (char *)0 );

list_set(
    button_container_list,
    widget_container_new() );

widget_container_new()->widget_button->button =
    button_back();

WIDGET_CONTAINER *widget_container_new(
    button, (char *)0 );

list_set(
    button_container_list,
    widget_container_new() );

widget_container_new()->widget_button->button =
    button_forward();

WIDGET_CONTAINER *widget_container_new(
    button, (char *)0 );

list_set(
    button_container_list,
    widget_container_new() );

widget_container_new()->widget_button->button =
    button_recall(
        load_javascript );

if ( post_choose_folder_action_string )
{
    WIDGET_CONTAINER *widget_container_new(
        button, (char *)0 );

    list_set(
        button_container_list,
        widget_container_new() );

    widget_container_new()->widget_button->button =
        button_restart_drillthru(
            post_choose_folder_action_string );
}

WIDGET_CONTAINER *widget_container_new(
    button, (char *)0 );

list_set(
    button_container_list,
    widget_container_new() );

widget_container_new()->widget_button->button =
    button_to_top();

WIDGET_CONTAINER *widget_container_new(
    table_close, (char *)0 ) );

list_set(
    button_container_list,
    widget_container_new() );
```

FORM_PROMPT_INSERT (4)

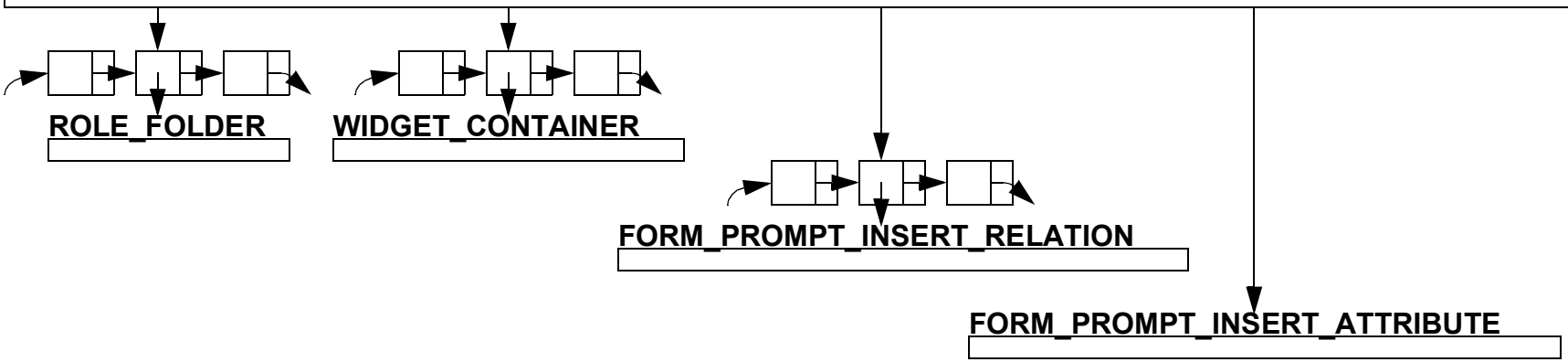
```
/* Usage */
char *form_prompt_insert_html(
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    LIST *form_prompt_insert_widget_list->
        form_prompt_insert_relation_list,
    char *form_tag(),
    LIST *form_prompt_insert_widget_list->
        widget_container_list,
    char *form_prompt_insert_button_list_html(),
    DICTIONARY *pair_dictionary,
    char *form_close_tag() );

/* Process */
char html[];
char *ptr = html;

ptr += sprintf( ptr,
    "%s\n%s",
    form_tag,

LIST *form_prompt_insert_relation_ajax_client_list(
    form_prompt_insert_relation_list );

if ( list_length(
        form_prompt_insert_relation_ajax_client_list() ) )
{
    char *ajax_client_list_javascript(
        session_key,
        login_name,
        role_name,
        form_prompt_insert_relation_ajax_client_list() );

    ptr += sprintf( ptr,
        "%s\n",
        ajax_client_list_javascript() );

    free( ajax_client_list_javascript() );
}

char *widget_container_list_html(
    APPASERVER_INSERT_STATE,
    JAVASCRIPT_PROMPT_ROW_NUMBER
        /* probably 0 */,
    (char *)0 /* background_color */,
    widget_container_list );

ptr += sprintf( ptr,
    "%s\n",
    widget_container_list_html() );

free( widget_container_list_html() );

char *widget_container_list_hidden_html(
    JAVASCRIPT_PROMPT_ROW_NUMBER
        /* probably 0 */,
    widget_container_list );

if ( widget_container_list_hidden_html() )
{
    ptr += sprintf( ptr,
        "%s\n",
        widget_container_list_hidden_html() );

    free( widget_container_list_hidden_html() );
}

if ( dictionary_length( pair_dictionary ) )
{
    char *dictionary_separate_hidden_html(
        DICTIONARY_SEPARATE_PAIR_PREFIX,
        pair_dictionary );

    ptr += sprintf( ptr,
        "%s\n",
        dictionary_separate_hidden_html() );

    free( dictionary_separate_hidden_html() );
}

ptr += sprintf( ptr,
    "%s\n%s",
    form_prompt_insert_button_list_html,
    form_close_tag );

char *form_prompt_insert_capacity_message(
    const char *__FUNCTION__ /* function_name */,
    const int STRING_704K /* max_strlen */,
    const double APPASERVER_CAPACITY_THRESHOLD,
    char *folder_name,
    int strlen( html ) /* strlen_html */ );

if ( form_prompt_insert_capacity_message() )
{
    appaserver_error_message(
        (char *)0 /* application_name */,
        (char *)0 /* login_name */,
        form_prompt_insert_capacity_message() );

    free( form_prompt_insert_capacity_message() );
}
```

FORM_PROMPT_INSERT_WIDGET_LIST (1)

```
/* Usage */
FORM_PROMPT_INSERT_WIDGET_LIST *
    form_prompt_insert_widget_list_new(
        const char *PROMPT_INSERT_LOOKUP_CHECKBOX,
        char *application_name,
        char *session_key,
        char *login_name,
        char *role_name,
        char *folder_name,
        char *post_change_javascript,
        LIST *folder_attribute_non_primary_name_list,
        LIST *relation_mto1_list(),
        LIST *relation_mto1_isa_list(),
        DICTIONARY *drillthru_dictionary,
        SECURITY_ENTITY *security_entity,
        LIST *folder_attribute_append_isa_list(),
        boolean role_folder_lookup_boolean,
        boolean form_prompt_insert_ignore_boolean() );

/* Process */
FORM_PROMPT_INSERT_WIDGET_LIST *
    form_prompt_insert_widget_list_calloc(
        void );

LIST *role_folder_insert_list( role_name );

LIST *widget_container_list = list_new();

list_set(
    widget_container_list,
    ( WIDGET_CONTAINER *widget_container =
            widget_container_new(
                table_open, (char *)0 ) ) );

widget_container->table_open->border_boolean = 1;

for FOLDER_ATTRIBUTE *folder_attribute in
    folder_attribute_append_isa_list
{
    if (    folder_attribute->omit_insert
    ||      folder_attribute->omit_insert_prompt )
    {
        continue;
    }

    ++tab_order;

    if ( ( FORM_PROMPT_INSERT_RELATION *
            form_prompt_insert_relation =
            form_prompt_insert_relation_new(
                application_name,
                session_key,
                login_name,
                role_name,
                folder_name /* many_folder_name */,
                post_change_javascript,
                folder_attribute_non_primary_name_list,
                relation_mto1_list
                    /* many_folder_relation_mto1_list */,
                relation_mto1_isa_list,
                drillthru_dictionary,
                security_entity,
                role_folder_insert_list(),
                form_prompt_insert_relation_list,
                folder_attribute->attribute_name,
                form_prompt_insert_ignore_boolean,
                tab_order ) ) )
    {
        if ( !form_prompt_insert_relation_list )
        {
            LIST *form_prompt_insert_relation_list =
                list_new();
        }

        list_set(
            form_prompt_insert_relation_list,
            form_prompt_insert_relation );

        list_set_list(
            widget_container_list,
            form_prompt_insert_relation->
                widget_container_list );

        continue;
    }

    FORM_PROMPT_INSERT_ATTRIBUTE *
    form_prompt_insert_attribute =
        form_prompt_insert_attribute_new(
            application_name,
            login_name,
            post_change_javascript,
            folder_attribute->attribute_name,
            folder_attribute->prompt,
            folder_attribute->attribute->datatype_name,
            folder_attribute->attribute->width,
            folder_attribute->attribute->hint_message,
            form_prompt_insert_ignore_boolean,
            form_prompt_insert_relation_list,
            tab_order );

    list_set_list(
        widget_container_list,
        form_prompt_insert_attribute->
            widget_container_list );
}

if ( role_folder_lookup_boolean )
{
    list_set_list(
        widget_container_list,
        LIST *form_prompt_insert_widget_list_lookup_list(
            const char *
                PROMPT_INSERT_LOOKUP_CHECKBOX ) );
}

list_set(
    widget_container_list,
    WIDGET_CONTAINER *widget_container_new(
        table_close, (char *)0 ) );

LIST *widget_hidden_container_list(
    DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
    drillthru_dictionary );

list_set_list(
    widget_container_list,
    widget_hidden_container_list() );
```

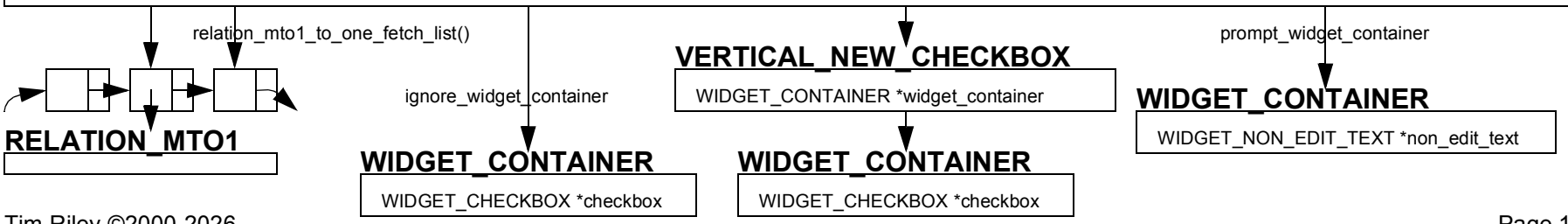
FORM_PROMPT_INSERT_WIDGET_LIST (2)

```
/* Attributes */
LIST *role_folder_insert_list;
LIST *widget_container_list;
LIST *form_prompt_insert_relation_list;
FORM_PROMPT_INSERT_ATTRIBUTE  *
    form_prompt_insert_attribute;
LIST *widget_hidden_container_list;
```



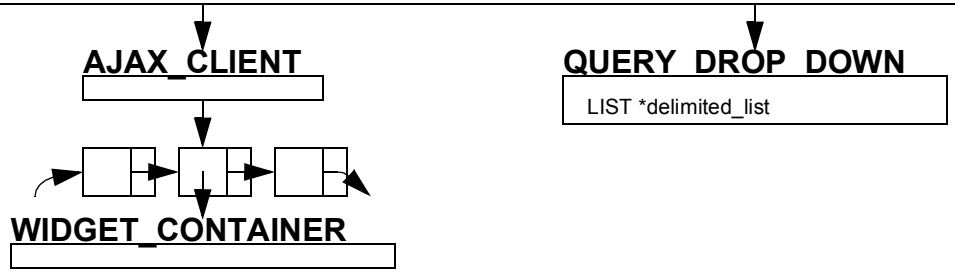
FORM_PROMPT_INSERT_RELATION (1)

```
/* Usage */
FORM_PROMPT_INSERT_RELATION *
    form_prompt_insert_relation_new(
        char *application_name,
        char *session_key,
        char *login_name,
        char *role_name,
        char *many_folder_name,
        char *post_change_javascript,
        LIST *folder_attribute_non_primary_name_list,
        LIST *many_folder_relation_mto1_list,
        LIST *relation_mto1_isa_list(),
        DICTIONARY *drillthru_dictionary,
        SECURITY_ENTITY *security_entity,
        LIST *role_folder_insert_list(),
        LIST *form_prompt_insert_relation_list,
        char *attribute_name,
        boolean form_prompt_insert_ignore_boolean(),
        int tab_order );

/* Process */
boolean form_prompt_insert_relation_attribute_exists(
    char *attribute_name,
    LIST *form_prompt_insert_relation_list );

if ( form_prompt_insert_relation_attribute_exists() )
        return NULL;

FORM_PROMPT_INSERT_RELATION *
    form_prompt_insert_relation_calloc(
        void );

RELATION_MTO1 *relation_mto1 =
    relation_mto1_consumes(
        attribute_name,
        many_folder_relation_mto1_list );

if ( !relation_mto1 )
{
    relation_mto1 =
        relation_mto1_isa_consumes(
            attribute_name,
            relation_mto1_isa_list );

    if ( !relation_mto1 ) return NULL;
}

LIST *widget_container_list = list_new();

list_set(
    widget_container_list,
    widget_container_new(
        table_row, (char *)0 ) );

if ( form_prompt_insert_ignore_boolean )
{
    list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );

    char *form_prefix_widget_name(
        char *FORM_PROMPT_INSERT_IGNORE_PREFIX,
        char *relation_mto1->relation_name );

    list_set(
        widget_container_list,
        ( ignore_widget_container =
            widget_container_new(
                checkbox,
                form_prefix_widget_name() ) ) );

    ignore_widget_container->
        checkbox->
        prompt =
            FORM_PROMPT_INSERT_IGNORE_PROMPT;

    ignore_widget_container->
        recall_boolean = 1;
}

list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );

boolean role_folder_insert_boolean(
    APPASERVER_INSERT_STATE,
    relation_mto1->one_folder_name,
    role_folder_insert_list );

if ( role_folder_insert_boolean )
{
    VERTICAL_NEW_CHECKBOX *
        vertical_new_checkbox =
            vertical_new_checkbox_new(
                VERTICAL_NEW_CHECKBOX_PREFIX,
                VERTICAL_NEW_CHECKBOX_PROMPT,
                relation_mto1->one_folder_name );

    list_set(
        widget_container_list,
        vertical_new_checkbox->
            widget_container );
}

list_set(
    widget_container_list,
    widget_container_new(
        table_data, (char *)0 ) );

list_set(
    widget_container_list,
    ( prompt_widget_container =
        widget_container_new(
            non_edit_text,
            relation_mto1->relation_prompt ) ) );
```



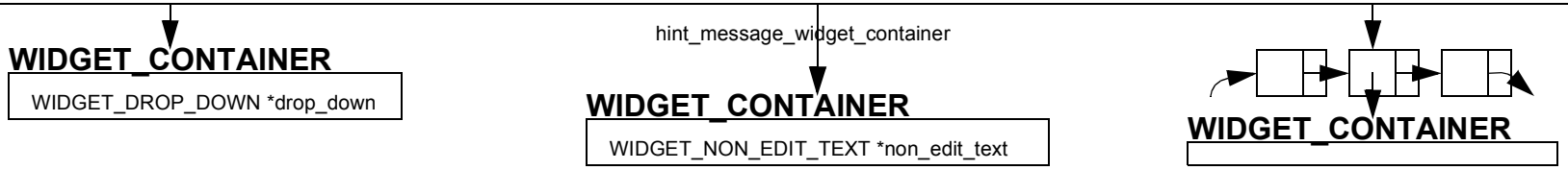
FORM_PROMPT_INSERT_RELATION (2)

```
/* Process (continued) */
list_set(
    widget_container_list,
    widget_container_new(
        table_data, (char *)0 ) );

LIST *relation_mto1_to_one_fetch_list(
    relation_mto1->one_folder_name,
    relation_mto1->
        one_folder->
        folder_attribute_primary_key_list );

if ( relation_mto1->
        relation->
        ajax_fill_drop_down )
{
    AJAX_CLIENT *ajax_client =
        /* Returns null if not participating */
        ajax_client_relation_mto1_new(
            relation_mto1,
            relation_mto1_to_one_fetch_list(),
            1 /* top_select_boolean */ );
}

if ( ajax_client )
{
    drop_down_widget_container =
        list_first(
            ajax_client->widget_container_list );

    drop_down_widget_container->
        drop_down->
        tab_order = tab_order;

    drop_down_widget_container->
        drop_down->
        post_change_javascript = post_change_javascript;

    drop_down_widget_container->
        recall_boolean = 1;

    list_set_list(
        widget_container_list,
        ajax_client->widget_container_list );

    goto form_prompt_insert_relation_hint_message;
}

list_set(
    widget_container_list,
    ( drop_down_widget_container =
        widget_container_new(
            drop_down,
            relation_mto1->relation_name
                /* widget_name */ ) ) );

LIST *relation_mto1_common_name_list(
    LIST *folder_attribute_non_primary_name_list,
    boolean relation_mto1->
        relation->
        copy_common_columns,
    LIST *relation_mto1->
        one_folder->
        folder_attribute_name_list );

QUERY_DROP_DOWN *query_drop_down_new(
    application_name,
    session_key,
    login_name,
    role_name,
    APPASERVER_INSERT_STATE,
    many_folder_name,
    relation_mto1->one_folder_name,
    relation_mto1->relation->related_attribute_name
        /* related_column */,
    relation_mto1->
        one_folder->
        folder_attribute_list,
    relation_mto1->
        one_folder->
        populate_drop_down_process_name,
    relation_mto1_to_one_fetch_list(),
    drillthru_dictionary
        /* drop_down_dictionary */,
    security_entity,
    relation_mto1_common_name_list() );
```



FORM_PROMPT_INSERT_RELATION (3)

```
/* Process (continued) */
LIST *widget_drop_down_option_list(
    ATTRIBUTE_MULTI_KEY_DELIMITER,
    WIDGET_DROP_DOWN_LABEL_DELIMITER,
    WIDGET_DROP_DOWN_EXTRA_DELIMITER,
    WIDGET_DROP_DOWN_DASH_DELIMITER,
    query_drop_down_new()->delimited_list,
    relation_mto1->
        one_folder->
        no_initial_capital );

drop_down_widget_container->
    drop_down->
    widget_drop_down_option_list =
        widget_drop_down_option_list();

drop_down_widget_container->
    drop_down->
    null_boolean = 1;

drop_down_widget_container->
    drop_down->
    top_select_boolean = 1;

drop_down_widget_container->
    drop_down->
    display_size = 1;

drop_down_widget_container->
    drop_down->
    tab_order = tab_order;

drop_down_widget_container->
    drop_down->
    post_change_javascript = post_change_javascript;

drop_down_widget_container->
    recall_boolean = 1;

form_prompt_insert_relation_hint_message:

if ( relation_mto1->relation->hint_message )
{
  list_set(
      widget_container_list,
      widget_container_new(
          table_data, (char *)0 ) );

  list_set(
      widget_container_list,
      ( hint_message_widget_container =
          widget_container_new(
              non_edit_text,
              relation_mto1->
                  relation->
                  hint_message ) ) );
}

/* Usage */
void form_prompt_insert_relation_javascript_save_set(
  LIST *form_prompt_insert_widget_list->
      form_prompt_insert_relation_list,
  char *widget_checkbox_submit_javascript() );

/* Process */
for FORM_PROMPT_INSERT_RELATION *
  form_prompt_insert_relation in
      form_prompt_insert_relation_list
{
  if ( form_prompt_insert_relation->
          vertical_new_checkbox )
  {
      form_prompt_insert_relation->
          vertical_new_checkbox->
          widget_container->
          checkbox->
          onclick =
              widget_checkbox_submit_javascript;
  }
}

/* Usage */
LIST *form_prompt_insert_relation_ajax_client_list(
  LIST *form_prompt_insert_relation_list );

/* Process */
LIST *ajax_client_list = {0};

for FORM_PROMPT_INSERT_RELATION *
  form_prompt_insert_relation in
      form_prompt_insert_relation_list
{
  if ( form_prompt_insert_relation->ajax_client )
  {
      if ( !ajax_client_list ) ajax_client_list = list_new();

      list_set(
          ajax_client_list,
          form_prompt_insert_relation->ajax_client );
  }
}

/* Attributes */
RELATION_MTO1 *relation_mto1;
WIDGET_CONTAINER *ignore_widget_container;
boolean role_folder_insert_boolean;
VERTICAL_NEW_CHECKBOX *vertical_new_checkbox;
LIST *relation_mto1_to_one_fetch_list;
AJAX_CLIENT *ajax_client;
LIST *relation_mto1_common_name_list;
QUERY_DROP_DOWN *query_drop_down;
LIST *widget_drop_down_option_list;
WIDGET_CONTAINER *prompt_widget_container;
WIDGET_CONTAINER *drop_down_widget_container;
WIDGET_CONTAINER *hint_message_widget_container;
LIST *widget_container_list;
```



FORM_PROMPT_INSERT_ATTRIBUTE (1)

```
/* Usage */
FORM_PROMPT_INSERT_ATTRIBUTE *
    form_prompt_insert_attribute_new(
        char *application_name,
        char *login_name,
        char *post_change_javascript,
        char *attribute_name,
        char *folder_attribute_prompt,
        char *datatype_name,
        int attribute_width,
        char *hint_message,
        boolean form_prompt_insert_ignore_boolean(),
        LIST *form_prompt_insert_relation_list,
        int tab_order );

/* Process */
boolean form_prompt_insert_relation_attribute_exists(
    attribute_name,
    form_prompt_insert_relation_list );

if ( form_prompt_insert_relation_attribute_exists()
        return NULL;

FORM_PROMPT_INSERT_ATTRIBUTE *
    form_prompt_insert_attribute_calloc(
        void );

LIST *widget_container_list = list_new();

list_set(
        widget_container_list,
        widget_container_new(
            table_row, (char *)0 ) );

if ( form_prompt_insert_ignore_boolean )
{
    list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );

    char *form_prefix_widget_name(
        char *FORM_PROMPT_INSERT_IGNORE_PREFIX,
        char *attribute_name );

    list_set(
        widget_container_list,
        ( ignore_widget_container =
            widget_container_new(
                checkbox,
                form_prefix_widget_name() ) ) );

    ignore_widget_container->
        checkbox->
        prompt =
            FORM_PROMPT_INSERT_IGNORE_PROMPT;
    ignore_widget_container->recall_boolean = 1;
}

/* Empty cell for “New” checkbox */
list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );

list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );

list_set(
    widget_container_list,
    ( prompt_widget_container =
        widget_container_new(
            non_edit_text,
            folder_attribute_prompt
                /* widget_name */ ) ) );

list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );
```

FORM_PROMPT_INSERT_ATTRIBUTE (2)

```
/* Process (continued) */
boolean attribute_is_notepad(
    ATTRIBUTE_DATATYPE_NOTEPAD,
    datatype_name );

if ( attribute_is_notepad() )
{
    list_set(
        widget_container_list,
        ( widget_container =
            widget_container_new(
                notepad,
                attribute_name
                    /* widget_name */ ) ) );

    widget_container->
        notepad->
        attribute_size = attribute_width;

    widget_container->
        notepad->
        tab_order = tab_order;

    widget_container->
        notepad->
        post_change_javascript = post_change_javascript;
}
else
if ( attribute_is_yes_no( attribute_name ) )
{
    list_set(
        widget_container_list,
        ( widget_container =
            widget_container_new(
                yes_no,
                attribute_name
                    /* widget_name */ ) ) );

    widget_container->
        yes_no->
        tab_order = tab_order;

    widget_container->
        yes_no->
        post_change_javascript = post_change_javascript;
}
else
if (    attribute_is_date( datatype_name )
||      attribute_is_date_time( datatype_name )
||      attribute_is_current_date_time( datatype_name ) )
{
    list_set(
        widget_container_list,
        ( widget_container =
            widget_container_new(
                widget_date,
                attribute_name
                    /* widget_name */ ) ) );

    widget_container->
        date->
        datatype_name = datatype_name;

    widget_container->
        date->
        attribute_width_max_length =
            attribute_width;

    widget_container->
        date->
        display_size =
            attribute_width;

    widget_container->
        date->
        tab_order = tab_order;

    widget_container->
        date->
        application_name = application_name;

    widget_container->
        date->
        login_name = login_name;

    widget_container->
        date->
        post_change_javascript = post_change_javascript;
}
else
if ( attribute_is_time( datatype_name ) )
{
    list_set(
        widget_container_list,
        ( widget_container =
            widget_container_new(
                widget_time,
                attribute_name
                    /* widget_name */ ) ) );

    widget_container->
        time->
        attribute_width_max_length =
            attribute_width;

    widget_container->
        time->
        widget_text_display_size =
            attribute_width;

    widget_container->
        time->
        tab_order = tab_order;

    widget_container->
        time->
        post_change_javascript = post_change_javascript;
}
else
```

FORM_PROMPT_INSERT_ATTRIBUTE (3)

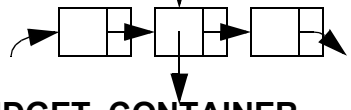
```
/* Process (continued) */
{
    list_set(
        widget_container_list,
        ( widget_container =
            widget_container_new(
                character,
                attribute_name
                    /* widget_name */ ) ) );

    widget_container->
        text->
        datatype_name = datatype_name;

    widget_container->
        text->
        attribute_width_max_length = attribute_width;

    int widget_text_display_size(
        WIDGET_TEXT_DEFAULT_MAX_LENGTH,
        attribute_width );

    widget_container->
        text->
        widget_text_display_size =
            widget_text_display_size();

    widget_container->
        text->
        tab_order = tab_order;

    widget_container->
        text->
        post_change_javascript = post_change_javascript;
}

boolean attribute_is_password(
    datatype_name );

if (    !attribute_is_notepad()
&&  !attribute_is_password() )
{
    widget_container->recall_boolean = 1;
}

if ( hint_message )
{
    list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );

    list_set(
        widget_container_list,
        ( hint_message_widget_container =
            widget_container_new(
                non_edit_text,
                hint_message
                    /* widget_name */ ) ) );

    hint_message_widget_container->
        non_edit_text->
        no_initial_capital_boolean = 1;
}

/* Attributes */
WIDGET_CONTAINER *ignore_widget_container;
WIDGET_CONTAINER *prompt_widget_container;
WIDGET_CONTAINER *widget_container;
WIDGET_CONTAINER *hint_message_widget_container;
LIST *widget_container_list;
boolean attribute_is_notepad;
boolean attribute_is_password;
```

ignore_widget_container

prompt_widget_container

widget_container

hint_message_widget_container



WIDGET_CONTAINER

WIDGET_CHECKBOX *checkbox

WIDGET_CONTAINER

WIDGET_NON_EDIT_TEXT *non_edit_text

WIDGET_CONTAINER

WIDGET_YES_NO *yes_no
WIDGET_TEXT *text
WIDGET_DATE *date
WIDGET_NOTEPAD *notepad

WIDGET_CONTAINER

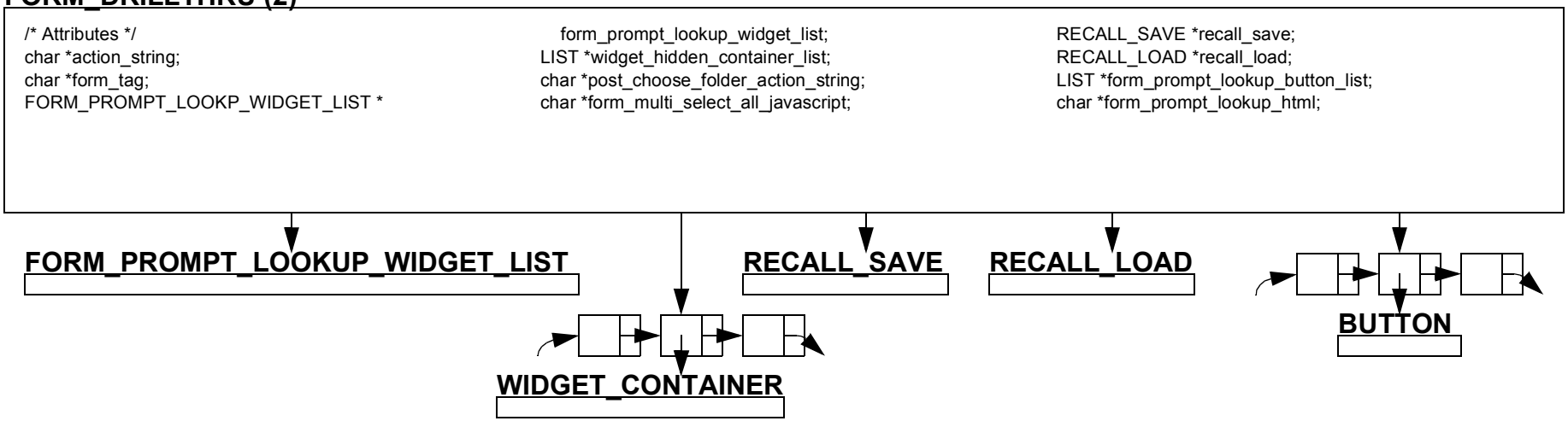
WIDGET_NON_EDIT_TEXT *non_edit_text

WIDGET_CONTAINER

FORM_DRILLTHRU (1)

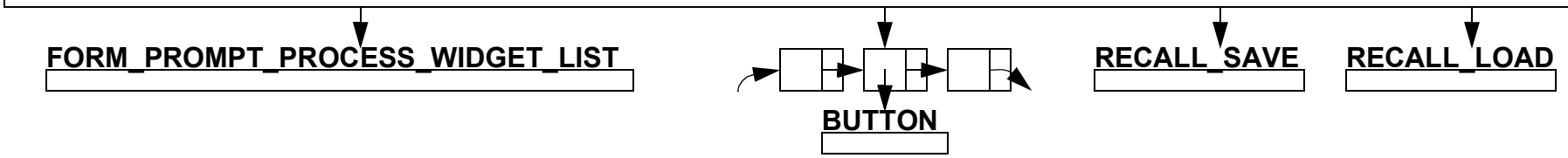
```
/* Usage */
FORM_DRILLTHRU *form_drillthru_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *state,
    char *base_folder_name,
    char *folder_name,
    boolean first_time_boolean,
    LIST *folder_attribute_list,
    LIST *relation_mto1_list,
    SECURITY_ENTITY *security_entity,
    DICTIONARY *drillthru_dictionary );

/* Process */
FORM_DRILLTHRU *form_drillthru_calloc( void );

char *form_drillthru_action_string(
    char *POST_DRILLTHRU_EXECUTABLE,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *state,
    char *folder_name );

char *form_tag(
        FORM_PROMPT_LOOKUP_NAME,
        form_drillthru_action_string(),
        FRAMESET_PROMPT_FRAME /* target_frame */ );

FORM_PROMPT_LOOKUP_WIDGET_LIST *
    form_prompt_lookup_widget_list =
        form_prompt_lookup_widget_list_new(
            application_name,
            session_key,
            login_name,
            role_name,
            folder_name /* many_folder_name */,
            relation_mto1_list,
            (LIST *)0 /* relation_mto1_isa_list */,
            (LIST *)0 /* relation_one2m_join_list */,
            folder_attribute_list,
            drillthru_dictionary,
            1 /* drillthru_status_active_boolean */,
            security_entity,
            0 /* not no_display_boolean */,
            0 /* not drop_down_extra_options_boolean */ );

list_set_first(
    widget_container_list,
    ( WIDGET_CONTAINER *widget_container =
            widget_container_new(
            table_open, (char *)0 ) ) );

widget_container->table_open->border_boolean = 1;

list_set(
    form_prompt_lookup_widget_list->
        widget_container_list,
    WIDGET_CONTAINER *widget_container_new(
        table_close, (char *)0 ) );

LIST *widget_hidden_container_list(
    DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
    drillthru_dictionary );

list_set_list(
    form_prompt_lookup_widget_list->
        widget_container_list,
        widget_hidden_container_list() );

list_set_list(
    form_prompt_lookup_widget_list->
        widget_container_list,
    widget_container_back_to_top_list() );

if ( !first_time_boolean )
{
    char *post_choose_folder_action_string(
        POST_CHOOSE_FOLDER_EXECUTABLE,
        application_name,
        session_key,
        login_name,
        role_name,
        APPASERVER_LOOKUP_STATE,
        base_folder_name );
}

char *form_multi_select_all_javascript(
    form_prompt_lookup_widget_list->
        widget_container_list );

RECALL_SAVE *recall_save =
    recall_save_new(
        APPASERVER_LOOKUP_STATE,
        FORM_PROMPT_LOOKUP_NAME,
        folder_name,
        form_prompt_lookup_widget_list->
            widget_container_list,
        application_ssl_support_boolean(
            application_name ) );

RECALL_LOAD *recall_load =
    recall_load_new(
        APPASERVER_LOOKUP_STATE,
        FORM_PROMPT_LOOKUP_NAME,
        folder_name,
        form_prompt_lookup_widget_list->
            widget_container_list );

boolean form_drillthru_skip_button_boolean(
    boolean first_time_boolean,
    char *state );

LIST *form_prompt_lookup_button_list(
    FORM_PROMPT_LOOKUP_NAME,
    form_drillthru_skip_button_boolean(),
    post_choose_folder_action_string(),
    form_multi_select_all_javascript(),
    recall_save->javascript,
    recall_load->javascript );

char *form_prompt_lookup_html(
    form_tag(),
    (LIST *)0 /* no_display_all_list */,
    (char *)0 /* radio_list_html */,
    (char *)0 /* form_prompt_lookup_ajax_javascript */,
    form_prompt_lookup_widget_list->
        widget_container_list,
    button_list_html(
        form_prompt_lookup_button_list() ),
    form_close_tag() );

free( button_list_html() );

button_list_free(
    form_prompt_lookup_button_list() );
```

FORM_DRILLTHRU (2)



FORM_PROMPT_PROCESS

```
/* Usage */
FORM_PROMPT_PROCESS *form_prompt_process_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *process_name,
    LIST *process_parameter_list,
    char *post_change_javascript,
    char *prompt_process_notepad(),
    boolean is_drillthru );

/* Process */
FORM_PROMPT_PROCESS *
    form_prompt_process_calloc(
        void );

FORM_PROMPT_PROCESS_WIDGET_LIST *
    form_prompt_process_widget_list =
        form_prompt_process_widget_list_new(
            application_name,
            login_name,
            process_parameter_list,
            post_change_process,
            prompt_process_notepad );

RECALL_SAVE *recall_save =
    recall_save_new(
        (char *)0 /* state */,
        FORM_PROMPT_PROCESS_NAME,
        (char *)0 /* folder_name */,
        form_prompt_process_widget_list->
            widget_container_list,
        application_ssl_support_boolean(
            application_name ) );

RECALL_LOAD *recall_load =
    recall_load_new(
        (char *)0 /* state */,
        FORM_PROMPT_PROCESS_NAME,
        (char *)0 /* folder_name */,
        form_prompt_process_widget_list->
            widget_container_list );

char *form_multi_select_all_javascript(
    form_prompt_process_widget_list->
        widget_container_list );

LIST *form_prompt_lookup_button_list(
    FORM_PROMPT_PROCESS_NAME,
    0 /* not button_drillthru_skip_boolean */,
    (char *)0 /* post_choose_folder_action_string */,
    form_multi_select_all_javascript(),
    recall_save->javascript,
    recall_load->javascript );

char *button_list_html(
    form_prompt_lookup_button_list() );

char *form_prompt_process_action_string(
    char *POST_PROMPT_PROCESS_EXECUTABLE,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *process_name,
    boolean is_drillthru );

const char *form_prompt_process_target_frame(
    const char *FRAMESET_PROMPT_FRAME,
    const char *FRAMESET_TABLE_FRAME,
    boolean is_drillthru );

char *form_tag(
    FORM_PROMPT_PROCESS_NAME,
    form_prompt_process_action_string(),
    form_prompt_process_target_frame() );

char *form_prompt_process_html(
    char *form_tag(),
    LIST *form_prompt_process_widget_list->
        widget_container_list,
    char *button_list_html(),
    char *form_close_tag() );

free( button_list_html() );

button_list_free(
    form_prompt_lookup_button_list() );

/* Attributes */
FORM_PROMPT_PROCESS_WIDGET_LIST *
    form_prompt_process_widget_list;
RECALL_SAVE *recall_save;
RECALL_LOAD *recall_load;
char *form_multi_select_all_javascript;
LIST *form_prompt_lookup_button_list;
char *button_list_html;
char *action_string;
const char *target_frame;
char *form_tag;
char *html;
```



FORM_PROMPT_PROCESS_WIDGET_LIST (1)

```
/* Usage */
FORM_PROMPT_PROCESS_WIDGET_LIST *
   form_prompt_process_widget_list_new(
      char *application_name,
      char *login_name,
      LIST *process_parameter_list,
      char *post_change_javascript );

/* Process */
LIST *widget_container_list = list_new();

FORM_PROMPT_PROCESS_WIDGET_LIST *
   form_prompt_process_widget_list_calloc(
      void );

list_set(
   widget_container_list(
      ( WIDGET_CONTAINER *widget_container =
         widget_container_new(
            table_open, (char *)0 ) ) );

   widget_container->
      table_open->
      border_boolean = 1;

if ( prompt_process_notepad )
{
   list_set_list(
      widget_container_list,
      form_prompt_process_widget_list_notepad_list(
         prompt_process_notepad ) );
}

for PROCESS_PARAMETER *process_parameter in
   process_parameter_list
{
   list_set(
      widget_container_list,
      widget_container_new(
         table_row, (char *)0 ) );

if ( process_parameter->
      process_parameter_yes_no )
{
   FORM_PROMPT_PROCESS_YES_NO *
      form_prompt_process_yes_no =
         form_prompt_process_yes_no_new(
            process_parameter->
               process_parameter_yes_no->
               prompt_name,
            process_parameter->
               process_parameter_yes_no->
               process_parameter_prompt->
               hint_message,
            post_change_process );

   list_set_set(
      widget_container_list,
      form_prompt_process_yes_no->
         form_prompt_lookup_yes_no->
         widget_container_list );
}
else
if ( process_parameter->
      process_parameter_attribute )
{
   FORM_PROMPT_ATTRIBUTE *
      form_prompt_attribute =
         form_prompt_attribute_new(
            application_name,
            login_name,
            process_parameter->
               process_parameter_attribute->
               attribute->
               attribute_name,
            process_parameter_attribute->
               attribute->
               attribute_name
                  /* folder_attribute_prompt */,
            process_parameter_attribute->
               attribute->
               datatype_name,
            process_parameter_attribute->
               attribute->
               width,
            process_parameter_attribute->
               attribute->
               hint_message );

      list_set_set(
         widget_container_list,
         form_prompt_attribute->
            widget_container_list );
}
else
if ( process_parameter->
      process_parameter_drop_down )
{
   FORM_PROMPT_PROCESS_DROP_DOWN *
      form_prompt_process_drop_down =
         form_prompt_process_drop_down_new(
            process_parameter->
               process_parameter_drop_down,
            post_change_javascript );

      list_set_list(
         widget_container_list,
         form_prompt_process_drop_down->
            widget_container_list );
}
else
```

FORM_PROMPT_PROCESS_YES_NO

```
FORM_PROMPT_LOOKUP_YES_NO *
      form_prompt_lookup_yes_no
```

FORM_PROMPT_ATTRIBUTE

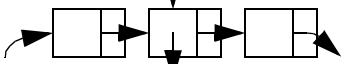
FORM_PROMPT_PROCESS_DROP_DOWN

FORM_PROMPT_PROCESS_WIDGET_LIST (2)

```
/* Process (continued) */
    if ( process_parameter->
           process_parameter_drop_down_prompt )
    {
        FORM_PROMPT_PROCESS_DROP_DOWN_PROMPT*
            form_prompt_process_drop_down_prompt =
            form_prompt_process_drop_down_prompt_new(
                  process_parameter->
                     process_parameter_drop_down_prompt,
                  post_change_javascript );

        list_set_list(
            widget_container_list,
            form_prompt_process_drop_down_prompt->
               widget_container_list );
    }
    else
    if ( process_parameter->
           process_parameter_prompt )
    {
        FORM_PROMPT_PROCESS_PROMPT *
            form_prompt_process_prompt =
                form_prompt_process_prompt_new(
                    application_name,
                    login_name,
                    process_parameter->
                       process_parameter_prompt->
                       prompt_name,
                    process_parameter->
                       process_parameter_prompt->
                       input_width,
                    process_parameter->
                       process_parameter_prompt->
                       hint_message,
                    process_parameter->
                       process_parameter_prompt->
                       upload_filename_boolean,
                    process_parameter->
                       process_parameter_prompt->
                       date_boolean,
                    post_change_javascript );

        list_set_list(
            widget_container_list,
                form_prompt_process_prompt->
                   widget_container_list );
    }

    list_set(
        widget_container_list,
        widget_container_new(
            table_close, (char *)0 ) );

    /* Attributes */
    FORM_PROMPT_PROCESS_YES_NO *
        form_prompt_process_yes_no;
    FORM_PROMPT_ATTRIBUTE *
        form_prompt_attribute;
    FORM_PROMPT_PROCESS_DROP_DOWN *
        form_prompt_process_drop_down;
    FORM_PROMPT_PROCESS_DROP_DOWN_PROMPT *
        form_prompt_process_drop_down_prompt;
    FORM_PROMPT_PROCESS_PROMPT *
        form_prompt_process_prompt;
    LIST *widget_container_list;
```

FORM_PROMPT_PROCESS_DROP_DOWN_PROMPT

FORM_PROMPT_PROCESS_PROMPT



WIDGET_CONTAINER

FORM_PROMPT_PROCESS_YES_NO

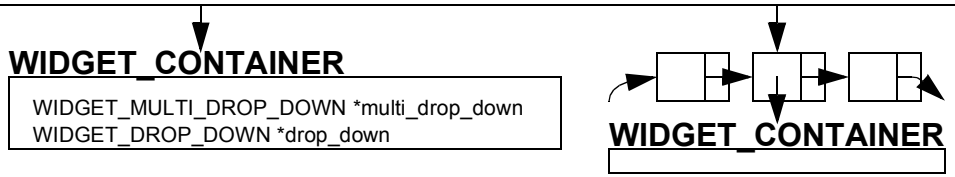
```
/* Usage */
FORM_PROMPT_PROCESS_YES_NO *
  form_prompt_process_yes_no_new(
      char *prompt_name,
      char *hint_message,
      char *post_change_javascript );

/* Process */
if ( !attribute_is_yes_no(
      prompt_name
      /* attribute_name */ ) )
{
  return NULL;
}

FORM_PROMPT_PROCESS_YES_NO *
  form_prompt_process_yes_no_calloc(
      void );

FORM_PROMPT_LOOKUP_YES_NO *
  form_prompt_lookup_yes_no =
      form_prompt_lookup_yes_no_new(
          prompt_name /* attribute_name */,
          prompt_name /* folder_attribute_prompt */,
          hint_message,
          post_change_javascript,
          0 /* not no_display_boolean */,
          (LIST *)0 /* form_prompt_lookup_relation_list */ );

/* Attributes */
FORM_PROMPT_LOOKUP_YES_NO *
  form_prompt_lookup_yes_no;
```

FORM_PROMPT_LOOKUP_YES_NO

```
LIST *widget_container_list
```

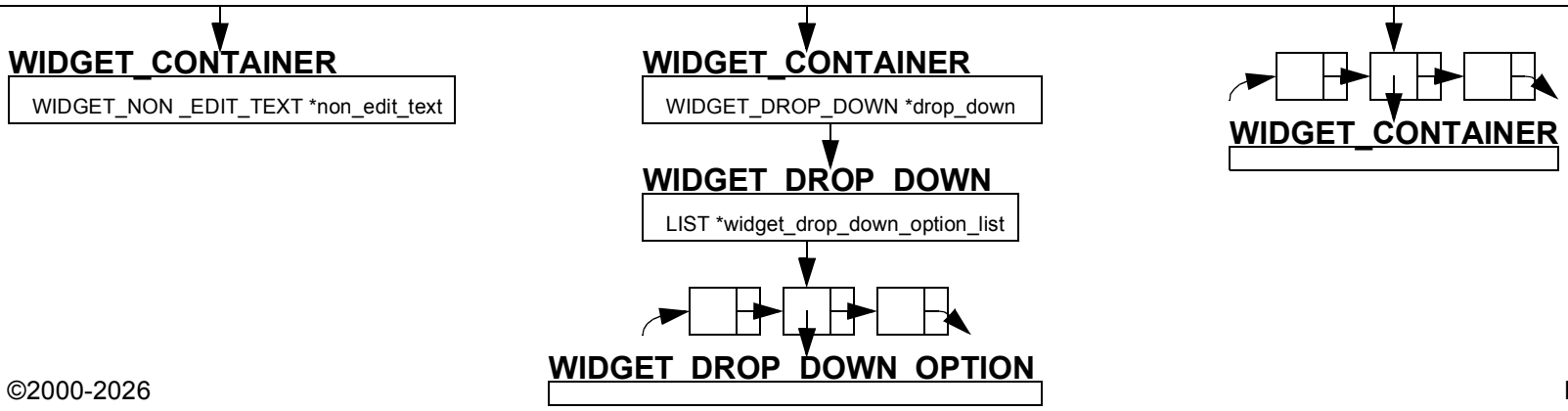
FORM_PROMPT_PROCESS_DROP_DOWN

```
/* Usage */
FORM_PROMPT_PROCESS_DROP_DOWN *
    form_prompt_process_drop_down_new(
        PROCESS_PARAMETER_DROP_DOWN *
            process_parameter_drop_down,
        char *post_change_javascript );

/* Process */
FORM_PROMPT_PROCESS_DROP_DOWN *
    form_prompt_process_drop_down_calloc(
        void );

LIST *widget_container_list = list_new();

list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );

char *form_prompt_process_drop_down_name(
    FOLDER *process_parameter_drop_down->folder,
    char *process_parameter_drop_down->name );

list_set(
    widget_container_list,
    widget_container_new(
        non_edit_text,
        form_prompt_process_drop_down_name()
            /* widget_name */ ) );

list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );

if ( process_parameter_drop_down->
        multi_select_boolean )
{
    list_set(
        widget_container_list,
        ( drop_down_widget_container =
            widget_container_new(
                multi_drop_down,
                process_parameter_drop_down->name
                    /* widget_name */ ) ) );

    drop_down_widget_container->
        multi_drop_down->
        original_drop_down->
        widget_drop_down_option_list =
            process_parameter_drop_down->
                widget_drop_down_option_list;

    drop_down_widget_container->
        multi_drop_down->
        original_drop_down->
        top_select_boolean  = 1;

    drop_down_widget_container->
        multi_drop_down->
        original_drop_down->
        display_size =
            WIDGET_MULTI_DISPLAY_SIZE;

    drop_down_widget_container->
        multi_drop_down->
        original_drop_down->
        post_change_javascript = post_change_javascript;
}
else
{
    list_set(
        widget_container_list,
        ( drop_down_widget_container =
            widget_container_new(
                drop_down,
                process_parameter_drop_down->name
                    /* widget_name */ ) ) );

    drop_down_widget_container->
        drop_down->
        widget_drop_down_option_list =
        process_parameter_drop_down->
            widget_drop_down_option_list;

    drop_down_widget_container->
        drop_down->
        top_select_boolean = 1;

    drop_down_widget_container->
        drop_down->
        display_size = 1;

    drop_down_widget_container->
        drop_down->
        post_change_javascript = post_change_javascript;
}

drop_down_widget_container->recall_boolean = 1;

/* Attributes */
WIDGET_CONTAINER *drop_down_widget_container;
LIST *widget_container_list;
```



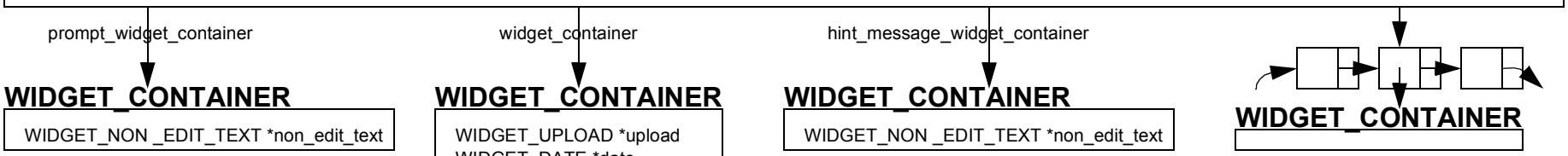
FORM_PROMPT_PROCESS_DROP_DOWN_PROMPT

```
/* Usage */
FORM_PROMPT_PROCESS_DROP_DOWN_PROMPT *
    form_prompt_process_drop_down_prompt_new(
        PROCESS_PARAMETER_DROP_DOWN_PROMPT *
            process_parameter_drop_down_prompt,
        char *post_change_javascript );

/* Process */
FORM_PROMPT_PROCESS_DROP_DOWN_PROMPT *
    form_prompt_process_drop_down_prompt_calloc(
        void );

LIST *widget_container_list = list_new();

list_set(
    widget_container_list,
    widget_container_new(
        table_data, (char *)0 ) );

char *form_prompt_process_drop_down_prompt_name(
    process_parameter_drop_down_prompt->
        drop_down_prompt_name,
    process_parameter_drop_down_prompt->
        optional_display );

list_set(
    widget_container_list,
    ( prompt_widget_container =
        widget_container_new(
            non_edit_text,
            form_prompt_process_drop_down_prompt_name() ) ) );

list_set(
    widget_container_list,
    widget_container_new(
        table_data, (char *)0 ) );

list_set(
    widget_container_list,
    ( drop_down_widget_container =
        widget_container_new(
            drop_down,
            form_prompt_process_drop_down_prompt_name() ) ) );

drop_down_widget_container->
    drop_down->
    widget_drop_down_option_list =
        process_parameter_drop_down_prompt->
            option_list;

drop_down_widget_container->
    drop_down->
    top_select_boolean = 1;

drop_down_widget_container->
    drop_down->
    multi_select_boolean =
        process_parameter_drop_down_prompt->
            multi_select_boolean;

drop_down_widget_container->
    drop_down->
    post_change_javascript =
        post_change_javascript;

drop_down_widget_container->recall_boolean = 1;

/* Attributes */
WIDGET_CONTAINER *prompt_widget_container;
WIDGET_CONTAINER *drop_down_widget_container;
LIST *widget_container_list;
```



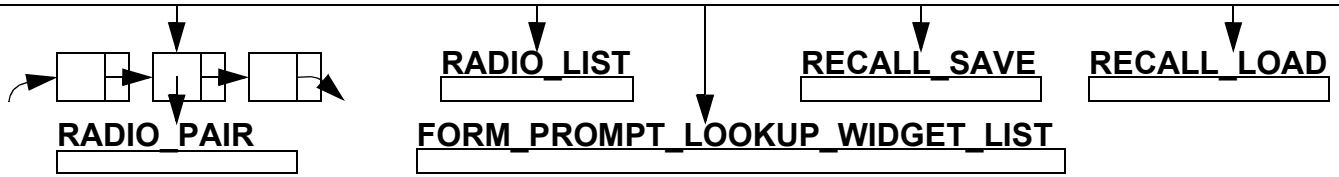
FORM_PROMPT_PROCESS_PROMPT

```
/* Usage */
FORM_PROMPT_PROCESS_PROMPT *
    form_prompt_process_prompt_new(
        char *application_name,
        char *login_name,
        char *prompt_name,
        int input_width,
        char *hint_message,
        boolean upload_filename_boolean,
        boolean date_boolean,
        char *post_change_javascript );

/* Process */
FORM_PROMPT_PROCESS_PROMPT *
    form_prompt_process_prompt_calloc(
        void );

LIST *widget_container_list = list_new();

list_set(
    widget_container_list,
    widget_container_new(
        table_data, (char *)0 ) );

list_set(
    widget_container_list,
    ( prompt_widget_container =
          widget_container_new(
              non_edit_text,
              prompt_name
                  /* widget_name */ ) ) );

list_set(
        widget_container_list,
            widget_container_new(
                table_data, (char *)0 ) );

if ( upload_filename_boolean )
{
    WIDGET_CONTAINER *widget_container =
        form_prompt_process_prompt_upload_widget(
            char *prompt_name,
            char *post_change_javascript );
}
else
if ( date_boolean )
{
    WIDGET_CONTAINER *widget_container =
        form_prompt_process_prompt_date_widget(
            char *application_name,
            char *login_name,
            char *prompt_name,
            char *post_change_javascript );
}
else
if ( input_width > 128 )
{
    WIDGET_CONTAINER *widget_container =
        form_prompt_process_prompt_notepad_widget(
            char *prompt_name,
            int input_width,
            char *post_change_javascript );
}
else
{
    int form_prompt_process_prompt_size(
        int input_width,
        int FORM_PROMPT_PROCESS_PROMPT_SIZE );

    WIDGET_CONTAINER *widget_container =
        form_prompt_process_prompt_text_widget(
            char *prompt_name,
            int input_width,
            int form_prompt_process_prompt_size(),
            char *post_change_javascript );
}

list_set( widget_container_list, widget_container );

if ( hint_message )
{
    list_set(
        widget_container_list,
            widget_container_new( table_data, (char *)0 ) );

    list_set(
        widget_container_list,
        ( hint_message_widget_container =
            widget_container_new(
                non_edit_text,
                hint_message
                    /* widget_name */ ) ) );

    hint_message_widget_container->
        non_edit_text->
        no_initial_capital_boolean = 1;
}

/* Attributes */
WIDGET_CONTAINER *prompt_widget_container;
WIDGET_CONTAINER *widget_container;
WIDGET_CONTAINER *hint_message_widget_container;
LIST *widget_container_list;
```



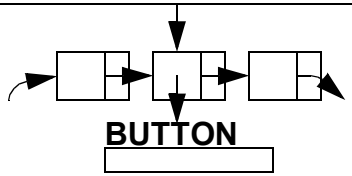
FORM_PROMPT_LOOKUP (1)

```
/* Usage */
FORM_PROMPT_LOOKUP *form_prompt_lookup_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    boolean prompt_lookup_omit_insert_button(),
    boolean prompt_lookup_omit_delete_button(),
    boolean prompt_lookup_include_chart_buttons(),
    boolean prompt_lookup_include_sort_button(),
    LIST *relation_mto1_list,
    LIST *relation_mto1_isa_list,
    LIST *relation_one2m_join_list,
    LIST *folder_attribute_append_isa_list,
    DICTIONARY *drillthru_dictionary,
    boolean drillthru_status_active_boolean,
    char *post_choose_folder_action_string() );

/* Process */
FORM_PROMPT_LOOKUP *form_prompt_lookup_calloc(
    void );

char *form_prompt_lookup_action_string(
    char *POST_PROMPT_LOOKUP_EXECUTABLE,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name );

char *form_tag(
    FORM_PROMPT_LOOKUP_NAME,
    form_prompt_lookup_action_string(),
    FRAMESET_TABLE_FRAME /* target_frame */ );

/* Note: creates a dangling table open tag */
LIST *form_prompt_lookup_no_display_all_widget_list();

LIST *form_prompt_lookup_radio_pair_list(
    prompt_lookup_omit_insert_button,
    prompt_lookup_omit_delete_button,
    prompt_lookup_include_chart_buttons,
    prompt_lookup_include_sort_button,
    list_length( relation_mto1_list )
        /* relation_mto1_list_length */ );

RADIO_LIST *radio_list =
    radio_list_new(
        FORM_RADIO_LIST_NAME,
        form_prompt_lookup_radio_pair_list(),
        RADIO_LOOKUP_VALUE /* initial_value */ );

FORM_PROMPT_LOOKUP_WIDGET_LIST *
    form_prompt_lookup_widget_list =
        form_prompt_lookup_widget_list_new(
            application_name,
            login_name,
            role_name,
            folder_name,
            relation_mto1_list,
            relation_mto1_isa_list,
            relation_one2m_join_list,
            folder_attribute_append_isa_list,
            drillthru_dictionary,
            drillthru_status_active_boolean,
            (SECURITY_ENTITY *)0 /* security_entity */,
            1 /* no_display_boolean */,
            1 /* drop_down_extra_options_boolean */ );

list_set(
    form_prompt_lookup_widget_list->
        widget_container_list,
    WIDGET_CONTAINER *widget_container_new(
        table_close, (char *)0 ) );

LIST *widget_hidden_container_list(
    DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
    drillthru_dictionary );

list_set_list(
    form_prompt_lookup_widget_list->
        widget_container_list,
        widget_hidden_container_list() );

list_set_list(
    form_prompt_lookup_widget_list->
        widget_container_list,
    widget_container_back_to_top_list() );

char *form_multi_select_all_javascript(
    form_prompt_lookup_widget_list->
        form_prompt_lookup_widget_list->
            widget_container_list );

char *form_prompt_lookup_ajax_javascript(
    session_key,
    login_name,
    role_name,
    form_prompt_lookup_widget_list->
        form_prompt_lookup_relation_list );

RECALL_SAVE *recall_save =
    recall_save_new(
        APPASERVER_LOOKUP_STATE,
        FORM_PROMPT_LOOKUP_NAME,
        folder_name,
        form_prompt_lookup_widget_list->
            widget_container_list,
        application_ssl_support_boolean(
            application_name ) );

RECALL_LOAD *recall_load =
    recall_load_new(
        APPASERVER_LOOKUP_STATE,
        FORM_PROMPT_LOOKUP_NAME,
        folder_name,
        form_prompt_lookup_widget_list->
            widget_container_list );
```



FORM_PROMPT_LOOKUP (2)

```
/* Process (continued) */
LIST *form_prompt_lookup_button_list(
    FORM_PROMPT_LOOKUP_NAME,
    0 /* not button_drillthru_skip_boolean */,
    post_choose_folder_action_string,
    form_multi_select_all_javascript(),
    recall->save_javascript,
    recall->load_javascript );

char *button_list_html(
    form_prompt_lookup_button_list() );

char *form_prompt_lookup_html(
    form_tag(),
    form_prompt_lookup_no_display_all_widget_list(),
    radio_list->html,
    form_prompt_lookup_ajax_javascript(),
    form_prompt_lookup_widget_list->
        widget_container_list,
    button_list_html(),
    form_close_tag() );

free( button_list_html() );

button_list_free(
    form_prompt_lookup_button_list() );

/* Attributes */
char *form_tag;
LIST *no_display_all_widget_list;
LIST *radio_pair_list;
RADIO_LIST *radio_list;
char *action_string;
FORM_PROMPT_LOOKUP_WIDGET_LIST *
    form_prompt_lookup_widget_list;
LIST *widget_container_hidden_list;
char *form_multi_select_all_javascript;
char *ajax_javascript;
RECALL_SAVE *recall_save;
RECALL_LOAD *recall_load;
LIST *button_list;
char *html;
```



FORM_PROMPT_LOOKUP (3)

```
/* Usage */
LIST *form_prompt_lookup_radio_pair_list(
    boolean prompt_lookup_omit_insert_button(),
    boolean prompt_lookup_omit_delete_button(),
    boolean prompt_lookup_include_chart_buttons(),
    boolean prompt_lookup_include_sort_button(),
    int relation_mto1_list_length );

/* Process */
LIST *radio_pair_list = list_new();

list_set(
    radio_pair_list,
    radio_pair_new(
        RADIO_STATISTICS_VALUE,
        RADIO_STATISTICS_DISPLAY ) );

if ( relation_mto1_list_length )
{
    list_set(
        radio_pair_list,
        radio_pair_new(
            RADIO_GROUP_COUNT_VALUE,
            RADIO_GROUP_COUNT_DISPLAY ) );
}

list_set(
    radio_pair_list,
    radio_pair_new(
        RADIO_SPREADSHEET_VALUE,
        RADIO_SPREADSHEET_DISPLAY ) );

if ( !prompt_lookup_omit_insert_button )
{
    list_set(
        radio_pair_list,
        radio_pair_new(
            RADIO_INSERT_VALUE,
            RADIO_INSERT_DISPLAY ) );
}

if ( !prompt_lookup_omit_delete_button )
{
    list_set(
        radio_pair_list,
        radio_pair_new(
            RADIO_DELETE_VALUE,
            RADIO_DELETE_DISPLAY ) );
}

if ( prompt_lookup_include_chart_buttons )
{
    list_set(
        radio_pair_list,
        radio_pair_new(
            RADIO_RCHART_VALUE,
            RADIO_RCHART_DISPLAY ) );

    list_set(
        radio_pair_list,
        radio_pair_new(
            RADIO_GRACE_VALUE,
            RADIO_GRACE_DISPLAY ) );

    list_set(
        radio_pair_list,
        radio_pair_new(
            RADIO_GOOGLE_VALUE,
            RADIO_GOOGLE_DISPLAY ) );

    list_set(
        radio_pair_list,
        radio_pair_new(
            RADIO_HISTOGRAM_VALUE,
            RADIO_HISTOGRAM_DISPLAY ) );
}

if ( prompt_lookup_include_sort_button )
{
    list_set(
        radio_pair_list,
        radio_pair_new(
            RADIO_SORT_VALUE,
            RADIO_SORT_DISPLAY ) );
}

list_set(
    radio_pair_list,
    radio_pair_new(
        RADIO_LOOKUP_VALUE,
        RADIO_LOOKUP_DISPLAY ) );
```

FORM_PROMPT_LOOKUP (4)

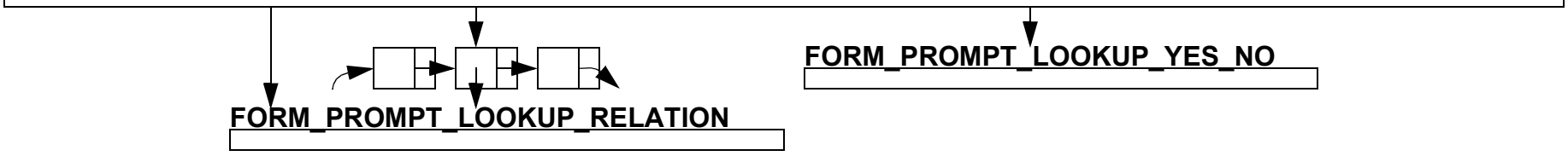
```
/* Usage */
LIST *form_prompt_lookup_join_container_list(
    char *DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
    LIST *relation_one2m_join_list );

/* Process */

/* Usage */
char *form_prompt_lookup_html(
    char *form_tag(),
    LIST *form_prompt_lookup->
        no_display_all_widget_list,
    char *radio_list_html,
    char *form_prompt_lookup_ajax_javascript,
    LIST *form_prompt_lookup_widget_list->
        widget_container_list,
    char *button_list_html(),
    char *form_close_tag() );

/* Process */
if ( list_length(
        form_prompt_lookup_no_display_all_widget_list ) )
{
    char *no_display_all_html =
    widget_container_list_html(
        (char *)0 /* state */,
        JAVASCRIPT_PROMPT_ROW_NUMBER
            /* probably 0 */,
        (char *)0 /* background_color */,
        form_prompt_lookup_no_display_all_widget_list );
}

char *container_list_html =
    widget_container_list_html(
        (char *)0 /* state */,
        JAVASCRIPT_PROMPT_ROW_NUMBER
            /* probably 0 */,
        (char *)0 /* background_color */,
        widget_container_list );

char *hidden_html =
    widget_container_list_hidden_html(
        JAVASCRIPT_PROMPT_ROW_NUMBER
            /* probably 0 */,
        widget_container_list );

snprintf(
    html[],
    sizeof ( html ),
    "%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s",
    form_tag,
    (form_prompt_lookup_ajax_javascript)
        ? form_prompt_lookup_ajax_javascript
        : "",
    button_list_html,
    (no_display_all_html) ? no_display_all_html : "",
    (radio_list_html) ? radio_list_html : "",
    container_list_html,
    (hidden_html) ? hidden_html : "",
    form_close_tag );

free( widget_container_list_html() );

if ( widget_container_list_hidden_html() )
    free( widget_container_list_hidden_html() );

/* Usage */
LIST *form_prompt_lookup_no_display_all_widget_list(
    void );

/* Process */
LIST *widget_container_list = list_new();

list_set(
    widget_container_list,
    ( WIDGET_CONTAINER *widget_container =
            widget_container_new(
            table_open, (char *)0 ) ) );

widget_container->table_open->border_boolean = 1;

list_set(
    widget_container_list,
    widget_container_new(
        table_row, (char *)0 ) );

list_set(
    widget_container_list,
    widget_container_new(
        table_data, (char *)0 ) );

list_set(
    widget_container_list,
    ( WIDGET_CONTAINER *widget_container =
            widget_container_new(
            checkbox, (char *)0 ) ) );

widget_container->
    checkbox->
    prompt =
        FORM_NO_DISPLAY_HEADING;

char *form_prompt_lookup_no_display_all_onclick(
    DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX );

widget_container->
    checkbox->
    onclick =
        form_prompt_lookup_no_display_all_onclick() );

list_set(
    widget_container_list,
    ( WIDGET_CONTAINER *widget_container =
            widget_container_new(
            table_data, (char *)0 ) ) );

widget_container->
    table_data->
    column_span = 99;

/* Usage */
char *form_prompt_lookup_no_display_all_onclick(
    char *FORM_NO_DISPLAY_PREFIX );

/* Process */
sprintf( static onclick[],
    "timlib_set_all_push_buttons( this, '%s' )",
    FORM_NO_DISPLAY_PREFIX );
```

FORM_PROMPT_LOOKUP (5)

```
/* Usage */
LIST *form_prompt_lookup_button_list(
    char *form_name,
    boolean button_drillthru_skip_boolean,
    char *post_choose_folder_action_string(),
    char *form_multi_select_all_javascript(),
    char *recall->save_javascript,
    char *recall->load_javascript );

/* Process */
LIST *button_list = list_new();

list_set(
    button_list,
    BUTTON *button_submit(
        form_name,
        form_multi_select_all_javascript,
        save_javascript,
        (char *)0
        /* form_verify_notepad_widths_javascript */ ) );

list_set(
    button_list,
    BUTTON *button_reset(
        form_name,
        (char *)0 /* javascript_replace */ ) );

list_set(
    button_list,
    button_back() );

list_set(
    button_list,
    button_forward() );

list_set(
    button_list,
    button_recall(
        load_javascript ) );

if ( post_choose_folder_action_string )
{
    list_set(
        button_list,
        button_restart_drillthru(
            post_choose_folder_action_string ) );
}

if ( button_drillthru_skip_boolean )
{
    list_set(
        button_list,
        button_drillthru_skip() );
}

/* Usage */
char *form_prompt_lookup_ajax_javascript(
    char *session_key,
    char *login_name,
    char *role_name,
    LIST *form_prompt_lookup_relation_list );

/* Process */
LIST *form_prompt_lookup_relation_ajax_client_list(
    LIST *form_prompt_lookup_relation_list );

if ( list_length(
        form_prompt_lookup_relation_ajax_client_list() ) )
{
    char *ajax_client_list_javascript(
        session_key,
        login_name,
        role_name,
        form_prompt_insert_relation_ajax_client_list() );
}
```

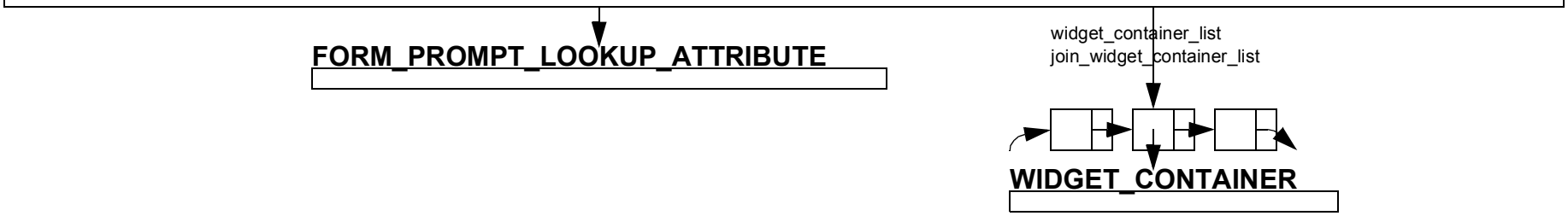
FORM_PROMPT_LOOKUP_WIDGET_LIST (1)

```
/* Usage */
FORM_PROMPT_LOOKUP_WIDGET_LIST *
    form_prompt_lookup_widget_list_new(
        char *application_name,
        char *session_key,
        char *login_name,
        char *role_name,
        char *folder_name,
        LIST *relation_mto1_list,
        LIST *relation_mto1_isa_list,
        LIST *relation_one2m_join_list(),
        LIST *folder_attribute_append_isa_list,
        DICTIONARY *drillthru_dictionary,
        boolean drillthru_status_active_boolean,
        SECURITY_ENTITY *security_entity,
        boolean no_display_boolean,
        boolean drop_down_extra_options_boolean );

/* Process */
FORM_PROMPT_LOOKUP_WIDGET_LIST *
    form_prompt_lookup_widget_list_calloc(
        void );

LIST *widget_container_list = list_new();

for FOLDER_ATTRIBUTE *folder_attribute in
    folder_attribute_append_isa_list
{
    list_set(
        widget_container_list,
    WIDGET_CONTAINER *widget_container_new(
        table_row, (char *)0 ) );

if ( ( FORM_PROMPT_LOOKUP_RELATION *
        form_prompt_lookup_relation =
            form_prompt_lookup_relation_new(
                application_name,
                session_key,
                login_name,
                role_name,
                folder_name
                    /* many_folder_name */,
                relation_mto1_list
                    /* many_folder_relation_mto1_list */,
                relation_mto1_isa_list,
                folder_attribute->attribute_name,
                folder_attribute->primary_key_index,
                drillthru_dictionary,
                drillthru_status_active_boolean,
                login_name,
                security_entity,
                no_display_boolean,
                drop_down_extra_options_boolean,
                form_prompt_lookup_relation_list ) ) )
{
    list_set(
        form_prompt_lookup_relation_list,
        form_prompt_lookup_relation );
    list_set_list(
        widget_container_list,
        form_prompt_lookup_relation->
            widget_container_list );

    continue;
}

if ( ( FORM_PROMPT_LOOKUP_YES_NO *
        form_prompt_lookup_yes_no =
            form_prompt_lookup_yes_no_new(
                folder_attribute->attribute_name,
                folder_attribute->prompt,
                folder_attribute->attribute->hint_message,
                (char *)0 /* post_change_javascript */,
                no_display_boolean,
                form_prompt_lookup_relation_list ) ) )
{
    list_set_list(
        widget_container_list,
        form_prompt_lookup_yes_no->
            widget_container_list );

    continue;
}
```



FORM_PROMPT_LOOKUP_WIDGET_LIST (2)

```
/* Process (continued) */
    if ( ( FORM_PROMPT_LOOKUP_ATTRIBUTE *
           form_prompt_lookup_attribute =
              form_prompt_lookup_attribute_new(
                 application_name,
                 login_name,
                 folder_attribute->attribute_name,
                 folder_attribute->prompt,
                 folder_attribute->
                    attribute->
                    datatype_name,
                 folder_attribute->attribute->width,
                 folder_attribute->attribute->hint_message,
                 no_display_boolean,
                 form_prompt_lookup_relation_list ) ) )
    {
            list_set_list(
               widget_container_list,
               form_prompt_lookup_attribute->
                  widget_container_list );
        }
    }

    if ( list_length( relation_one2m_join_list ) )
    {
        LIST *join_widget_container_list =
            LIST *form_prompt_lookup_join_container_list(
                DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
                relation_one2m_join_list );

        list_set_list(
            widget_container_list,
                join_widget_container_list );
    }

    void widget_containter_list_validate_date_unset(
        widget_container_list /* in/out */ );

    /* Attributes */
    LIST *widget_container_list;
    FORM_PROMPT_LOOKUP_RELATION *
        form_prompt_lookup_relation;
    LIST *form_prompt_lookup_relation_list;
    FORM_PROMPT_LOOKUP_YES_NO *
        form_prompt_lookup_yes_no;
    FORM_PROMPT_LOOKUP_ATTRIBUTE *
        form_prompt_lookup_attribute;
    LIST *join_widget_container_list;
```



FORM_PROMPT_LOOKUP_RELATION (1)

```
/* Usage */
FORM_PROMPT_LOOKUP_RELATION *
    form_prompt_lookup_relation_new(
        char *application_name,
        char *login_name,
        char *role_name,
        char *many_folder_name,
        LIST *many_folder_relation_mto1_list,
        LIST *relation_mto1_isa_list,
        char *attribute_name,
        int primary_key_index,
        DICTIONARY *drillthru_dictionary,
        boolean drillthru_status_active_boolean,
        SECURITY_ENTITY *security_entity,
        boolean no_display_boolean,
        boolean drop_down_extra_options_boolean,
        LIST *form_prompt_lookup_relation_list );

/* Process */
if (    !list_length( many_folder_relation_mto1_list )
&&  !list_length( relation_mto1_isa_list ) )
{
    return NULL;
}

if ( form_prompt_lookup_relation_attribute_exists(
    char *attribute_name,
    LIST *form_prompt_lookup_relation_list ) )
{
    return NULL;
}

FORM_PROMPT_LOOKUP_RELATION *
    form_prompt_lookup_relation_calloc(
        void );

RELATION_MTO1 *relation_mto1 =
    relation_mto1_consumes(
        attribute_name,
        many_folder_relation_mto1_list );

if ( !relation_mto1 )
{
    relation_mto1 =
        relation_mto1_isa_consumes(
            attribute_name,
            relation_mto1_isa_list );

    if ( !relation_mto1 ) return NULL;
}

if (    primary_key_index
&&  relation_mto1->
            relation->
            relation_type_isa )
{
    return NULL;
}

LIST *widget_container_list = list_new();

if ( no_display_boolean )
{
    list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );

    char *form_prefix_widget_name(
        DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
        relation_mto1->relation_name
            /* attribute_name */ );

    list_set(
        widget_container_list,
        ( no_display_widget_container =
            widget_container_new(
                checkbox,
                form_prefix_widget_name() ) ) );

    no_display_widget_container->
        checkbox->
        prompt =
            FORM_NO_DISPLAY_HEADING;

    no_display_widget_container->recall_boolean = 1;
}

list_set(
    widget_container_list,
    widget_container_new(
        table_data, (char *)0 ) );

list_set(
    widget_container_list,
    ( prompt_widget_container =
        widget_container_new(
            non_edit_text,
            relation_mto1->relation_prompt ) ) );

list_set(
    widget_container_list,
    widget_container_new(
        table_data, (char *)0 ) );
```

RELATION_MTO1

no_display_widget_container

WIDGET_CONTAINER

WIDGET_CHECKBOX *checkbox

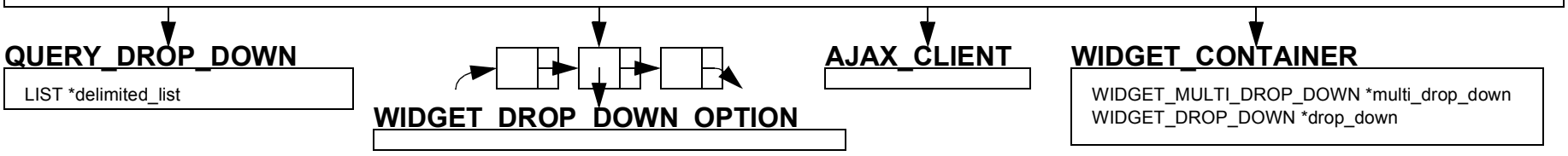
prompt_widget_container

WIDGET_CONTAINER

WIDGET_NON _EDIT_TEXT *non_edit_text

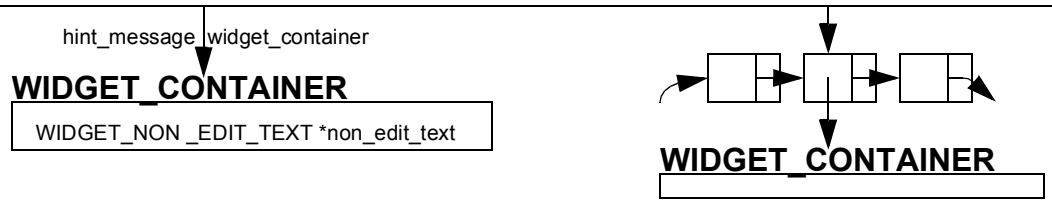
FORM_PROMPT_LOOKUP_RELATION (2)

```
/* Process (continued) */
if ( !relation_mto1->relation->omit_drillthru )
{
    LIST *relation_mto1_to_one_list =
        relation_mto1_list(
            relation_mto1->
                one_folder_name
                    /* many_folder_name */,
            relation_mto1->
                one_folder->
                folder_attribute_primary_key_list
                    /* many_folder_primary_key_list */ );

    QUERY_DROP_DOWN *query_drop_down_new(
        application_name,
        session_key,
        login_name,
        role_name,
        APPASERVER_LOOKUP_STATE,
        many_folder_name,
        relation_mto1->one_folder->folder_name
            /* one_folder_name */,
        relation_mto1->relation->related_attribute_name
            /* related_column */,
        relation_mto1->
            one_folder->
            folder_attribute_list,
        relation_mto1->
            one_folder->
            populate_drop_down_process_name,
        relation_mto1_to_one_list,
        drillthru_dictionary
            /* drop_down_dictionary */,
        security_entity,
        (LIST *)0 /* common_name_list */ );

    LIST *widget_drop_down_option_list(
        ATTRIBUTE_MULTI_KEY_DELIMITER,
        WIDGET_DROP_DOWN_LABEL_DELIMITER /* '|' */,
        WIDGET_DROP_DOWN_EXTRA_DELIMITER /* '[' */,
        WIDGET_DROP_DOWN_DASH_DELIMITER,
        query_drop_down_new()->delimited_list,
        relation_mto1->
            one_folder->
            no_initial_capital );
}

if ( relation_mto1->relation->drop_down_multi_select )
{
    list_set(
        widget_container_list,
        ( drop_down_widget_container =
            widget_container_new(
                multi_drop_down,
                relation_mto1->relation_name ) );

    drop_down_widget_container->
        multi_drop_down->
        original_drop_down->
        widget_drop_down_option_list =
            widget_drop_down_option_list();

    drop_down_widget_container->
        multi_drop_down->
        original_drop_down->
        display_size =
            WIDGET_MULTI_DISPLAY_SIZE;

    if ( drop_down_extra_options_boolean )
    {
        drop_down_widget_container->
            multi_drop_down->
            original_drop_down->
            null_boolean  = 1;

        drop_down_widget_container->
            multi_drop_down->
            original_drop_down->
            not_null_boolean = 1;
    }

    drop_down_widget_container->recall_boolean = 1;

    goto form_prompt_lookup_relation_hint_message;
}

AJAX_CLIENT *
    form_prompt_lookup_relation_ajax_client(
        role_name,
        relation_mto1 );

if ( form_prompt_lookup_relation()->ajax_client )
{
    drop_down_widget_container =
        list_first(
            form_prompt_lookup_relation_ajax_client()->
                widget_container_list );

    drop_down_widget_container->recall_boolean = 1;

    list_set_list(
        widget_container_list,
        form_prompt_lookup_relation_ajax_client()->
            widget_container_list );

    goto form_prompt_lookup_relation_hint_message;
}
```



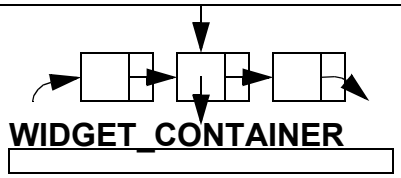
FORM_PROMPT_LOOKUP_RELATION (3)

```
/* Process (continued) */
list_set(
    widget_container_list,
    ( drop_down_widget_container =
        widget_container_new(
            drop_down,
            relation_mto1->relation_name ) ) );

drop_down_widget_container->
    drop_down->
    widget_drop_down_option_list =
        widget_drop_down_option_list();

drop_down_widget_container->
    drop_down->
    display_size = 1;

/* De Morgan's law: */
/* If active_boolean && drillthru, then skip */
if (    !drillthru_status_active_boolean
||      !relation_mto1->one_folder->drillthru )
{
    drop_down_widget_container->
        drop_down->
        top_select_boolean = 1;

    if ( drop_down_extra_options_boolean )
    {
        drop_down_widget_container->
            drop_down->
            null_boolean = 1;

        drop_down_widget_container->
            drop_down->
            not_null_boolean = 1;
    }
}
else
{
    drop_down_widget_container->
        drop_down->
        bottom_select_boolean = 1;
}

drop_down_widget_container->recall_boolean = 1;

form_prompt_lookup_relation_hint_message:

if ( relation_mto1->relation->hint_message )
{
    list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );

    list_set(
        widget_container_list,
        ( hint_message_widget_container =
            widget_container_new(
                non_edit_text,
                relation_mto1->relation->hint_message ) ) );
}

/* Usage */
AJAX_CLIENT *
    form_prompt_lookup_relation_ajax_client(
        char *role_name,
        RELATION_MTO1 *relation_mto1 );

/* Process */
LIST *relation_mto1_to_one_fetch_list(
    relation_mto1->one_folder_name,
    relation_mto1->
        one_folder->
        folder_attribute_primary_key_list );

if ( relation_mto1->
        relation->
        ajax_fill_drop_down )
{
    AJAX_CLIENT *ajax_client =
        /* Returns null if not participating */
        ajax_client_relation_mto1_new(
            relation_mto1,
            relation_mto1_to_one_fetch_list(),
            1 /* top_select_boolean */ );
}

/* Attributes */
RELATION_MTO1 *relation_mto1;
LIST *widget_container_list;
WIDGET_CONTAINER *no_display_widget_container;
WIDGET_CONTAINER *prompt_widget_container;
QUERY_DROP_DOWN *query_drop_down;
LIST *widget_drop_down_option_list;
WIDGET_CONTAINER *drop_down_widget_container;
AJAX_CLIENT *ajax_client;
WIDGET_CONTAINER *hint_message_widget_container;
```



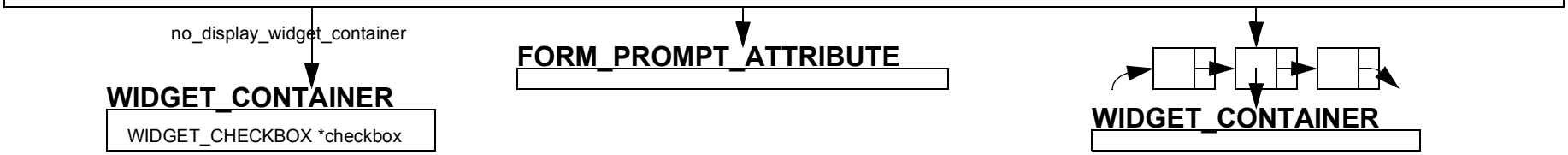
FORM_PROMPT_LOOKUP_YES_NO

```
/* Usage */
FORM_PROMPT_LOOKUP_YES_NO *
    form_prompt_lookup_yes_no_new(
        char *attribute_name,
        char *folder_attribute_prompt,
        char *hint_message,
        char *post_change_javascript,
        boolean no_display_boolean,
        LIST *form_prompt_lookup_relation_list );

/* Process */
if ( form_prompt_lookup_relation_attribute_exists(
    attribute_name,
    form_prompt_lookup_relation_list ) )
{
    return NULL;
}

if ( !attribute_is_yes_no( attribute_name ) )
{
    return NULL;
}

FORM_PROMPT_LOOKUP_YES_NO *
    form_prompt_lookup_yes_no_calloc(
        void );

LIST *widget_container_list = list_new();

if ( no_display_boolean )
{
    list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );

    char *form_prefix_widget_name(
        DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
        attribute_name );

    list_set(
        widget_container_list,
        ( no_display_widget_container =
            WIDGET_CONTAINER *widget_container_new(
                checkbox,
                form_prefix_widget_name() ) ) );

    no_display_widget_container->checkbox->prompt =
        FORM_NO_DISPLAY_HEADING;

    no_display_widget_container->
        recall_boolean = 1;
}

list_set(
    widget_container_list,
    widget_container_new(
        table_data, (char *)0 ) );

list_set(
    widget_container_list,
    ( prompt_widget_container =
        widget_container_new(
            non_edit_text,
            folder_attribute_prompt ) ) );

list_set(
    widget_container_list,
    widget_container_new(
        table_data, (char *)0 ) );

list_set(
    widget_container_list,
    ( yes_no_widget_container =
        widget_container_new(
            yes_no,
            attribute_name ) ) );

yes_no_widget_container->
    yes_no->
    post_change_javascript =
        post_change_javascript;

yes_no_widget_container->recall_boolean = 1;

if ( hint_message )
{
    list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );

    list_set(
        widget_container_list,
        ( hint_message_widget_container =
            widget_container_new(
                non_edit_text, hint_message
                    /* widget_name */ ) ) );

    hint_message_widget_container->
        non_edit_text->
        no_initial_capital_boolean = 1;
}

/* Attribute */
LIST *widget_container_list;
```



FORM_PROMPT_LOOKUP_ATTRIBUTE

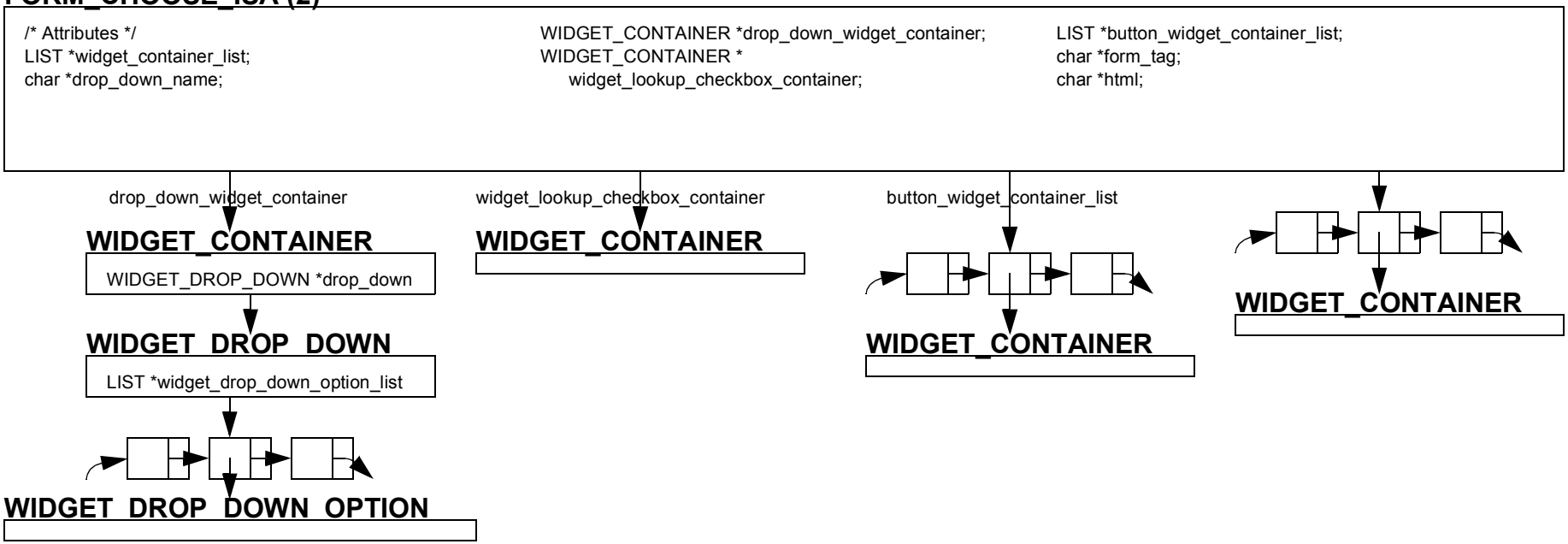
```
/* Usage */
FORM_PROMPT_LOOKUP_ATTRIBUTE *
    form_prompt_lookup_attribute_new(
        char *application_name,
        char *login_name,
        char *attribute_name,
        char *folder_attribute_prompt,
        char *datatype_name,
        int attribute_width,
        char *hint_message,
        boolean no_display_boolean,
        LIST *form_prompt_lookup_relation_list );

/* Process */
if ( form_prompt_lookup_relation_attribute_exists(
    attribute_name,
    form_prompt_lookup_relation_list ) )
{
    return NULL;
}

FORM_PROMPT_LOOKUP_ATTRIBUTE *
    form_prompt_lookup_attribute_calloc(
        void );

LIST *widget_container_list = list_new();

if ( no_display_boolean )
{
    list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );

    char *form_prefix_widget_name(
        DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
        attribute_name );

    list_set(
        widget_container_list,
        ( no_display_widget_container =
            WIDGET_CONTAINER *widget_container_new(
                checkbox,
                form_prefix_widget_name() ) ) );

    no_display_widget_container->checkbox->prompt =
        FORM_NO_DISPLAY_HEADING;

    no_display_widget_container->
        recall_boolean = 1;
}

FORM_PROMPT_ATTRIBUTE *
    form_prompt_attribute =
        form_prompt_attribute_new(
            application_name,
            login_name,
            attribute_name,
            folder_attribute_prompt,
            datatype_name,
            attribute_width,
            hint_message );

list_set_list(
    widget_container_list,
    form_prompt_attribute->
        widget_container_list );

/* Attributes */
LIST *widget_container_list;
WIDGET_CONTAINER *no_display_widget_container;
FORM_PROMPT_ATTRIBUTE *
    form_prompt_attribute;
```



FORM_CHOOSE_ISA (1)

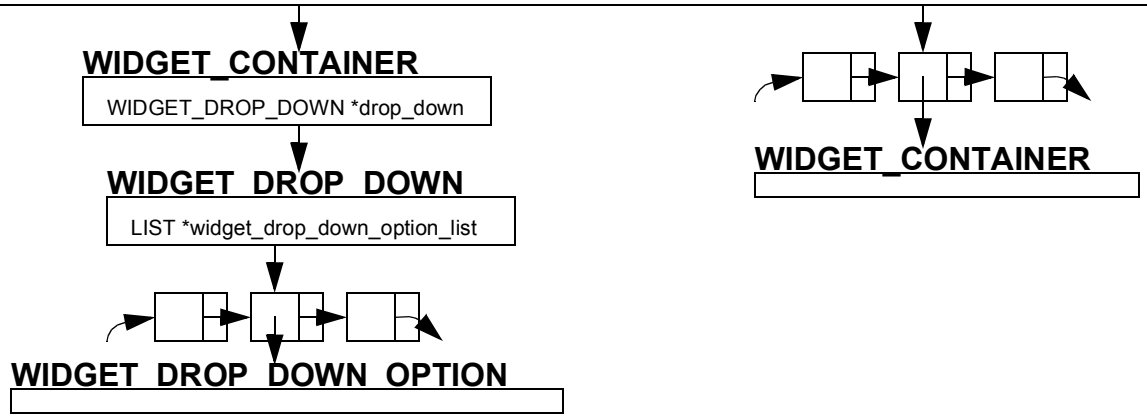
```
/* Usage */
FORM_CHOOSE_ISA *form_choose_isa_new(
    const char *WIDGET_LOOKUP_CHECKBOX_NAME,
    char *choose_isa_prompt_message(),
    LIST *primary_key_list,
    LIST *delimited_list,
    boolean no_initial_capital,
    char *choose_isa_action_string(),
    char *target_frame );

/* Process */
FORM_CHOOSE_ISA *form_choose_isa_calloc(
    void );

LIST *widget_container_list = list_new();

WIDGET_CONTAINER *widget_container_new(
     table_open, (char *)0 );

widget_container_new()->
    table_open->
    border_boolean = 1;

list_set(
  widget_container_list,
  WIDGET_CONTAINER *widget_container_new() );

list_set(
  widget_container_list,
  WIDGET_CONTAINER *widget_container_new(
     table_row, (char *)0 ) );

list_set(
  widget_container_list,
     WIDGET_CONTAINER *widget_container_new(
     table_data (char *)0 ) );

list_set(
  widget_container_list,
    widget_container_new(
        non_edit_text,
        choose_isa_prompt_mesage ) );

list_set(
  widget_container_list,
     WIDGET_CONTAINER *widget_container_new(
        table_data, (char *)0 ) );

char *form_choose_isa_drop_down_name(
    const char ATTRIBUTE_MULTI_KEY_DELIMITER,
    LIST *primary_key_list );

list_set(
  widget_container_list,
  ( drop_down_widget_container =
     WIDGET_CONTAINER *widget_container_new(
        drop_down,
        form_choose_isa_drop_down_name()
           /* widget_name */ ) ) );

drop_down_widget_container->
    drop_down->
    widget_drop_down_option_list =
      LIST *widget_drop_down_option_list(
        ATTRIBUTE_MULTI_KEY_DELIMITER,
        WIDGET_DROP_DOWN_LABEL_DELIMITER,
        WIDGET_DROP_DOWN_EXTRA_DELIMITER,
        WIDGET_DROP_DOWN_DASH_DELIMITER,
        delimited_list,
        no_initial_capital );

drop_down_widget_container->
  drop_down->
  top_select_boolean = 1;

list_set(
  widget_container_list,
  WIDGET_CONTAINER *widget_container_new(
     table_row, (char *)0 ) );

list_set(
  widget_container_list,
     WIDGET_CONTAINER *widget_container_new(
     table_data, (char *)0 ) );

WIDGET_CONTAINER *
    widget_lookup_checkbox_container(
        WIDGET_LOOKUP_CHECKBOX_NAME );

list_set(
    widget_container_list,
    widget_lookup_checkbox_container() );

list_set(
  widget_container_list,
  WIDGET_CONTAINER *widget_container_new(
        table_close, (char *)0 ) );

LIST *form_choose_isa_button_widget_container_list(
    void );

list_set_list(
    widget_container_list,
    form_choose_isa_button_widget_container_list() );

char *form_tag(
    FORM_CHOOSE_ISA_NAME,
    choose_isa_action_string,
    target_frame );

char *form_choose_isa_html(
    char *form_tag(),
    LIST *widget_container_list,
    char *form_close_tag() );
```

FORM_CHOOSE_ISA (2)



FORM_CHOOSE_ROLE

```
/* Usage */
FORM_CHOOSE_ROLE *form_choose_role_new(
    /* List may be all the roles or missing the current */
    LIST *role_name_list,
    char *post_choose_role_action_string(),
    char *target_frame /* FRAMESET_MENU_FRAME */,
    char *FORM_CHOOSE_ROLE_NAME,
    char *CHOOSE_ROLE_DROP_DOWN_NAME );

/* Process */
FORM_CHOOSE_ROLE *form_choose_role_calloc( void );

LIST *widget_container_list = list_new();

list_set(
    widget_container_list,
    widget_container_new(
        table_open, (char *)0 ) );

list_set(
    widget_container_list,
    widget_container_new(
        table_row, (char *)0 ) );

list_set(
    widget_container_list,
    widget_container_new(
        table_data, (char *)0 ) );

list_set(
    widget_container_list,
    ( WIDGET_CONTAINER *
        drop_down_widget_container =
            widget_container_new(
                drop_down,
                CHOOSE_ROLE_DROP_DOWN_NAME ) ) );

drop_down_widget_container->
    drop_down->
    widget_drop_down_option_list =
        LIST *widget_drop_down_option_list(
            ATTRIBUTE_MULTI_KEY_DELIMITER,
            WIDGET_DROP_DOWN_LABEL_DELIMITER,
            WIDGET_DROP_DOWN_EXTRA_DELIMITER,
            WIDGET_DROP_DOWN_DASH_DELIMITER,
            role_name_list,
            0 /* not no_initial_capital */ );

drop_down_widget_container->
    drop_down->
    top_select_boolean = 1;

char *form_choose_role_onchange_javascript(
    FORM_CHOOSE_ROLE_NAME );

drop_down_widget_containter->
    drop_down->
    post_change_javascript =
        form_choose_role_onchange_javascript();

list_set(
    widget_container_list,
    widget_container_new(
        table_close, (char *)0 ) );

char *form_tag(
    FORM_CHOOSE_ROLE_NAME,
    post_choose_role_action_string,
    target_frame );

char *form_choose_role_html(
    char *form_tag(),
    LIST *widget_container_list,
    char *form_close_tag() );

/* Usage */
char *form_choose_role_onchange_javascript(
    char *FORM_CHOOSE_ROLE_NAME );

/* Process */
sprintf(
    static onchange_javascript[],
    "%s.submit()",
    FORM_CHOOSE_ROLE_NAME );

/* Attributes */
LIST *widget_container_list;
WIDGET_CONTAINER *drop_down_widget_container;
char *onchange_javascript;
char *form_tag;
char *html;
```

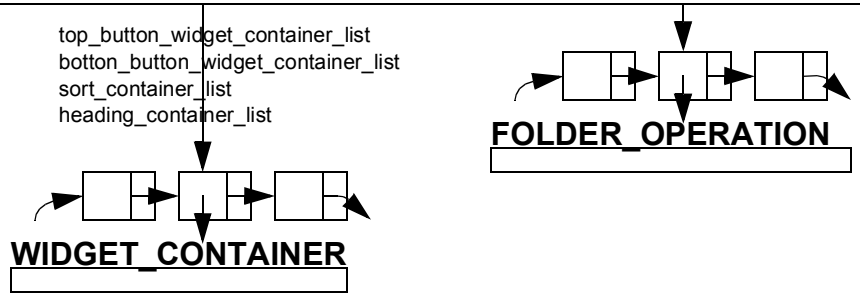


FORM_TABLE_EDIT (1)

```
/* Usage */
FORM_TABLE_EDIT *form_table_edit_new(
    char *folder_name,
    char *target_frame,
    pid_t process_id,
    char *folder->post_change_javascript,
    DICTIONARY *query_dictionary,
    DICTIONARY *drillthru_dictionary,
    DICTIOANRY *no_display_dictionary,
    LIST *folder_operation_list,
    int table_edit_query_row_list_length,
    char *table_edit_post_action_string,
    LIST *table_edit_heading_label_list,
    LIST *table_edit_heading_name_list,
    boolean
    table_edit_omit_heading_delete_checkbox_boolean );

/* Process */
FORM_TABLE_EDIT *form_table_edit_calloc(
    void );

int form_increment_number(
    process_id );

char *form_name(
    FORM_TABLE_EDIT_LABEL,
    form_increment_number() );

LIST *top_button_widget_container_list =
    form_table_edit_button_widget_container_list(
        form_name(),
        post_change_javascript,
        0 /* not button_to_top_boolean */ );

if ( table_edit_query_row_list_length > 10 )
{
    LIST *bottom_button_widget_container_list =
        form_table_edit_button_widget_container_list(
            form_name(),
            post_change_javascript,
            1 /* button_to_top_boolean */ );
}

boolean form_table_edit_sort_buttons_boolean(
    int FORM_TABLE_EDIT_MINIMUM_SORT,
    int table_edit_query_row_list_length );

if ( form_table_edit_sort_buttons_boolean() )
{
    LIST *form_table_edit_sort_container_list(
        char *form_name(),
        int list_length( folder_operation_list )
            /* folder_operation_list_length */,
        LIST *table_edit_heading_name_list );
}

char *query_dictionary_hidden_html =
    dictionary_separate_hidden_html(
        DICTIONARY_SEPARATE_QUERY_PREFIX,
        query_dictionary );

char *drillthru_dictionary_hidden_html =
    dictionary_separate_hidden_html(
        DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
        drillthru_dictionary );

char *no_display_dictionary_hidden_html =
    dictionary_separate_hidden_html(
        DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
        no_display_dictionary );

char *form_tag(
    form_name(),
    table_edit_post_action_string,
    target_frame );
```

FORM_TABLE_EDIT (2)

```
/* Process (continued) */
LIST *form_table_edit_heading_container_list(
    no_display_dictionary,
    folder_operation_list,
    table_edit_heading_label_list,
    form_name(),
    table_edit_omit_heading_delete_checkbox_boolean );

char *form_table_edit_heading_container_string(
    LIST *form_table_edit_heading_container_list() );

char *widget_table_open_tag(
    1 /* border_boolean */,
    0 /* cell_spacing */,
    0 /* cell_padding */ ),

char *form_table_edit_open_html(
    char *form_tag(),
    LIST *top_button_widget_container_list,
    char *widget_table_open_tag(),
    LIST *form_table_edit_sort_container_list(),
    char *form_table_edit_heading_container_string() );

char *form_table_edit_close_html(
    LIST *bottom_button_widget_container_list,
    char *query_dictionary_hidden_html,
    char *drillthru_dictionary_hidden_html,
    char *no_display_dictionary_hidden_html,
    char *form_close_tag() );

if ( list_length( folder_operation_list ) )
{
    LIST *form_table_edit_blank_folder_operation_list(
        list_length( folder_operation_list )
            /* folder_operation_list_length */ );

    LIST *form_table_edit_heading_container_list(
        no_display_dictionary,
        form_table_edit_blank_folder_operation_list(),
        table_edit_heading_label_list,
        form_name(),
        table_edit_omit_heading_delete_checkbox_boolean );

    free( form_table_edit_heading_container_string() );
    char *form_table_edit_heading_container_string(
        LIST *form_table_edit_heading_container_list() );
}

/* Attributes */
int form_increment_number;
char *form_name;
LIST *top_button_widget_container_list;
LIST *bottom_button_widget_container_list;
boolean sort_buttons_boolean;
LIST *sort_container_list;
LIST *heading_container_list;
char *heading_container_string;
char *query_dictionary_hidden_html;
char *drillthru_dictionary_hidden_html;
char *no_display_dictionary_hidden_html;
char *form_tag;
char *open_html;
char *close_html;
LIST *blank_folder_operation_list;
```



FORM_TABLE_EDIT (3)

```
/* Usage */
LIST *form_table_edit_button_widget_container_list(
    char *form_name(),
    char *post_change_javascript,
    boolean button_to_top_boolean );

/* Process */
LIST *widget_container_list = list_new();

list_set(
    widget_container_list,
    WIDGET_CONTAINER *widget_container_new(
        table_open, (char *)0 /* widget_name */ ) );

list_set(
    widget_container_list,
    WIDGET_CONTAINER *widget_container_new(
        table_row, (char *)0 /* widget_name */ ) );

list_set(
    widget_container_list,
    WIDGET_CONTAINER *widget_container_new(
        table_data, (char *)0 ) );

list_set(
    widget_container_list,
    ( WIDGET_CONTAINER *widget_container =
        widget_container_new(
            button, (char *)0 ) ) );

widget_container->widget_button->button =
    button_submit(
        form_name,
        (char *)0 /* multi_select_all_javascript */,
        (char *)0 /* recall_save_javascript */,
        (char *)0 /* verify_attribute_widths_javascript */ );

list_set(
    widget_container_list,
    ( WIDGET_CONTAINER *widget_container =
        widget_container_new(
            button, (char *)0 ) ) );

char *javascript_replace(
    post_change_javascript,
    APPASERVER_UPDATE_STATE,
    0 /* row_number */ );

widget_container->widget_button->button =
    button_reset(
        form_name,
        javascript_replace() );

list_set(
    widget_container_list,
    ( WIDGET_CONTAINER *widget_container =
        widget_container_new(
            button, (char *)0 ) ) );

widget_container->widget_button->button =
    button_back();

list_set(
    widget_container_list,
    ( WIDGET_CONTAINER *widget_container =
        widget_container_new(
            button, (char *)0 ) ) );

widget_container->widget_button->button =
    button_forward();

if ( button_to_top_boolean )
{
    list_set(
        widget_container_list,
        ( WIDGET_CONTAINER *widget_container =
            widget_container_new(
                button, (char *)0 ) ) );

    widget_container->widget_button->button =
        button_to_top();
}

list_set(
    widget_container_list,
    WIDGET_CONTAINER *widget_container_new(
        table_close, (char *)0 /* widget_name */ ) );

/* Usage */
LIST *form_table_edit_blank_folder_operation_list(
    int folder_operation_list_length );

/* Process */
LIST *folder_operation_list = list_new();

for (  int i = 0;
       i < folder_operation_list_length;
       i++ )
{
    FOLDER_OPERATION *folder_operation_calloc();

    folder_operation_calloc()->operation =
        OPERATION *operation_blank_new();

    list_set(
        folder_operation_list,
        folder_operation_calloc() );
}
```

FORM_TABLE_EDIT (4)

```
/* Usage */
LIST *form_table_edit_sort_container_list(
    char *folder_name,
    int list_length( folder_operation_list )
        /* folder_operation_list_length */,
    LIST *table_edit_heading_name_list );

/* Process */
LIST *widget_container_list = list_new();

snprintf(
    static onclick[],
    sizeof ( onclick ),
    "%s && push_button_submit('%s');",
    JAVASCRIPT_WAIT_START,
    folder_name );

/* Sort ascend row */

list_set(
    widget_container_list,
    widget_container_new(
        table_row, (char *)0 ) );

/* Placeholders for the operations */
for(   int i = 0;
       i < folder_operation_list_length;
       i++ )
{
    list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );
}

for char *heading_name in
    table_edit_heading_name_list
{
    list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );

    char *form_table_edit_heading_name(
        SQL_DELIMITER,
        FORM_SORT_ASCEND_LABEL,
        heading_name );

    snprintf(
        widget_name[],
        sizeof ( widget_name ),
        "%s%s",
        DICTIONARY_SEPARATE_SORT_PREFIX,
        form_table_edit_heading_name() );

    list_set(
        widget_container_list,
        ( widget_container =
            widget_container_new(
                checkbox,
                strdup( widget_name ) ) ) );

    widget_container->
        checkbox->
        prompt = "Sort";

    widget_container->
        checkbox->
        onclick = onclick;
}

/* Sort descend row */

list_set(
    widget_container_list,
    widget_container_new(
        table_row, (char *)0 ) );

/* Placeholders for the operations */
for(   int i = 0;
       i < folder_operation_list_length;
       i++ )
{
    list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );
}

for char *heading_name in
    table_edit_heading_name_list
{
    list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );

    char *form_table_edit_heading_name(
        SQL_DELIMITER,
        FORM_SORT_DESCEND_LABEL,
        heading_name );

    snprintf(
        widget_name[],
        sizeof ( widget_name ),
        "%s%s",
        DICTIONARY_SEPARATE_SORT_PREFIX,
        form_table_edit_heading_name() );

    list_set(
        widget_container_list,
        ( widget_container =
            widget_container_new(
                checkbox,
                strdup( widget_name ) ) ) );

    widget_container->
        checkbox->
        prompt = "Descend";

    widget_container->
        checkbox->
        onclick = onclick;
}
```

FORM_TABLE_EDIT (5)

```
/* Usage */
char *form_table_edit_heading_name(
    char SQL_DELIMITER,
    char *form_sort_label,
    char *heading_name );

/* Process */
static char heading_name[];
char *ptr = heading_name;

for (  int p = 0;
       piece(
          piece_buffer[],
          SQL_DELIMITER,
          heading_name,
          p );
       p++ )
{
    if ( ptr != heading_name )
    {
        ptr += sprintf( ptr, "%c", SQL_DELIMITER );
    }

    sprintf( name_buffer[],
        "%s%s",
        form_sort_label,
        piece_buffer );

    ptr += sprintf( ptr,
        "%s",
        name_buffer );
}

/* Usage */
LIST *form_table_edit_heading_container_list(
    DICTIONARY *no_display_dictionary,
    LIST *folder_operation_list,
    LIST *table_edit_heading_label_list,
    char *form_name,
    boolean
    table_edit_omit_heading_delete_checkbox_boolean );

/* Process */
LIST *widget_container_list = list_new();

for FOLDER_OPERATION *folder_operation in
    folder_operation_list
{
    LIST *form_table_edit_checkbox_container_list(
        form_name,
        table_edit_omit_delete_checkbox_boolean,
        folder_operation );

    list_set_list(
        widget_container_list,
        form_table_edit_checkbox_container_list() );
}

for char *heading_label in
    table_edit_heading_label_list
{
    boolean dictionary_key_boolean(
        no_display_dictionary,
        heading_label /* key */ );

    if ( dictionary_key_boolean() ) continue;

    char *string_initial_capital(
        heading_string[],
        heading_label );

    WIDGET_CONTAINER *widget_container =
        widget_container_new(
            table_heading,
            strdup( heading_string ) );

    list_set(
        widget_container_list,
        widget_container );
}
```

FORM_TABLE_EDIT (6)

```
/* Usage */
WIDGET_CONTAINER *
    form_table_edit_heading_checkbox_container(
        char *form_name,
        char *operation_name,
        WIDGET_CONTAINER *input_widget_container,
        char *delete_warning_javascript );

/* Process */
if ( input_widget_container->widget_type == blank )
{
    WIDGET_CONTAINER *widget_container =
        widget_container_new(
            blank,
            (char *)0 /* widget_name );

    return this;
}

WIDGET_CONTAINER *widget_container =
    widget_container_new(
        checkbox,
        operation_name
            /* widget_name */ );

char *form_table_edit_heading_checkbox_string(
    char *operation_name,
    char *input_widget_container->checkbox->prompt );

widget_container->
    checkbox->
    prompt =
        form_table_edit_heading_checkbox_string();

if ( delete_warning_javascript )
{
    snprintf(
        onclick[],
        sizeof ( onclick ),
        "%s && form_push_button_set_all('%s','%s');",
        delete_warning_javascript,
        form_name,
        operation_name );
}
else
{
    snprintf(
        onclick[],
        sizeof ( onclick ),
        "form_push_button_set_all('%s','%s');",
        form_name,
        operation_name );
}

widget_container->
    checkbox->
    onclick = strdup( onclick );

/* Usage */
LIST *form_table_edit_checkbox_container_list(
    char *form_name,
    boolean
    table_edit_omit_heading_delete_checkbox_boolean,
    FOLDER_OPERATION *folder_operation );

/* Process */
int string_strncmp(
        folder_operation->
            operation->
            operation_name,
        "delete" );

if (    table_edit_omit_heading_delete_checkbox_boolean
&&  string_strncmp() == 0 )
{
    char *string_initial_capital(
        buffer[],
        folder_operation->
            operation->
            operation_name );

    widget_container_new(
        table_heading,
        string_initial_capital() );

    list_set(
        widget_container_list,
        widget_container_new() );
}
else
{
    list_set(
        widget_container_list,
        widget_container_new(
            table_heading, (char *)0 ) );

    WIDGET_CONTAINER *
        form_table_edit_heading_checkbox_container(
            form_name,
            folder_operation->
                operation->
                operation_name,
            folder_operation->
                operation->
                widget_container,
            folder_operation->
                operation->
                delete_warning_javascript );

    list_set(
        widget_container_list,
        form_table_edit_heading_widget_container() );
}
```

PROMPT_LOOKUP (1)

```
/* Usage */
PROMPT_LOOKUP *prompt_lookup_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    boolean application_menu_horizontal_boolean,
    char *appaserver_parameter->data_directory,
    POST_DICTIONARY *post_dictionary );

/* Process */
PROMPT_LOOKUP *prompt_lookup_calloc(
    void );

PROMPT_LOOKUP_INPUT *prompt_lookup_input =
    prompt_lookup_input_new(
        application_name,
        session_key,
        login_name,
        role_name,
        folder_name,
        application_menu_horizontal_boolean,
        data_directory,
        post_dictionary->original_post_dictionary );

if ( prompt_lookup_input->forbid ) exit( 1 );

boolean relation_mto1_exists_multi_select(
    prompt_lookup_input->relation_mto1_list );

boolean prompt_lookup_omit_insert_button(
    boolean prompt_lookup_input->
        role_folder_insert_boolean,
    boolean prompt_lookup_input->
        drillthru_status->
        skipped_boolean,
    boolean relation_mto1_exists_multi_select() );

boolean prompt_lookup_omit_delete_button(
    LIST *prompt_lookup_input->folder_operation_list,
    ROW_SECURITY_ROLE_UPDATE_LIST *
        row_security_role_update_list );

boolean prompt_lookup_include_chart_buttons(
    LIST *prompt_lookup_input->
        folder_attribute_append_isa_list );

boolean prompt_lookup_include_sort_button(
    char *ATTRIBUTE_NAME_SORT_ORDER,
    char *ATTRIBUTE_NAME_DISPLAY_ORDER,
    char *ATTRIBUTE_NAME_SEQUENCE_NUMBER,
    LIST *prompt_lookup_input->
        folder->
        folder_attribute_name_list );

if ( prompt_lookup_input->
        drillthru_status->
        participating_boolean )
{
    char *post_choose_folder_action_string(
        POST_CHOOSE_FOLDER_EXECUTABLE,
        application_name,
        session_key,
        login_name,
        role_name,
        APPASERVER_LOOKUP_STATE,
        prompt_lookup_input->
            drillthru_status->
            base_folder_name );
}

FORM_PROMPT_LOOKUP *form_prompt_lookup =
    form_prompt_lookup_new(
        application_name,
        session_key,
        login_name,
        role_name,
        folder_name,
        prompt_lookup_omit_insert_button(),
        prompt_lookup_omit_delete_button(),
        prompt_lookup_include_chart_buttons(),
        prompt_lookup_include_sort_button(),
        prompt_lookup_input->relation_mto1_list,
        prompt_lookup_input->relation_mto1_isa_list,
        prompt_lookup_input->relation_one2m_join_list,
        prompt_lookup_input->
            folder_attribute_append_isa_list,
        prompt_lookup_input->
            dictionary_separate->
            drillthru_dictionary,
        prompt_lookup_input->
            drillthru_status->
            active_boolean,
            post_choose_folder_action_string() );

int folder_attribute_calendar_date_name_list_length(
    prompt_lookup_input->
        folder_attribute_append_isa_list );

char *document_head_calendar_setup_string(
    folder_attribute_date_name_list_length() );

char *application_title_string( application_name );

char *prompt_lookup_title_string(
    char *APPASERVER_LOOKUP_STATE,
    char *folder_name );

DOCUMENT *document =
    document_new(
        application_name,
        application_title_string(),
        prompt_lookup_title_string(),
        (char *)0 /* sub_title_string */,
        (char *)0 /* sub_sub_title_string */,
        prompt_lookup_input->
            folder->
            notepad,
        (char *)0 /* onload_javascript_string */,
        (prompt_lookup_input->menu)
            ? prompt_lookup_input>menu->html : (char *)0,
        document_head_menu_setup_string(
            (prompt_lookup_input->menu)
                ? 1 : 0 /* menu_boolean */ ),
        document_head_calendar_setup_string(),
        javascript_include_string(
            application_name,
            (LIST *)0 /* javascript_filename_list */ ) );

char *document_form_html(
    document->html,
    document->document_head->html,
    document_head_close_tag(),
    document->document_body->html,
    form_prompt_lookup->html,
    document_body_close_tag(),
    document_close_tag() );

free( form_prompt_lookup->html );
```

PROMPT_LOOKUP (2)

```
/* Driver */
printf(
    "%s\n",
    prompt_lookup->document_form_html );

/* Attributes */
PROMPT_LOOKUP_INPUT *prompt_lookup_input;
boolean relation_mto1_exists_multi_select;
boolean omit_insert_button;
boolean omit_delete_button;
boolean include_chart_buttons;
boolean include_sort_button;
char *post_choose_folder_action_string;
FORM_PROMPT_LOOKUP *form_prompt_lookup;
int folder_attribute_calendar_date_name_list_length;
char *application_title_string;
char *title_string;
DOCUMENT *document;
char *document_form_html;
```

PROMPT_LOOKUP_INPUT

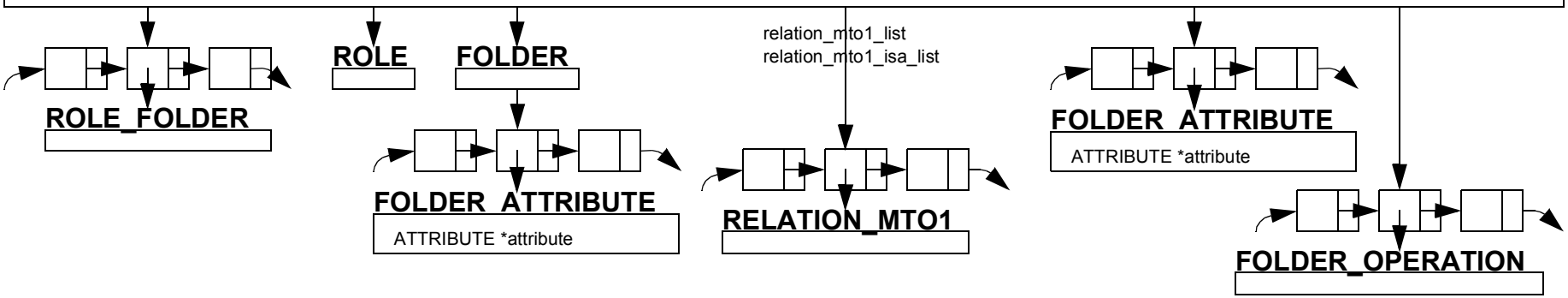
FORM_PROMPT_LOOKUP

DOCUMENT

DOCUMENT_HEAD *document_head
DOCUMENT_BODY *document_body

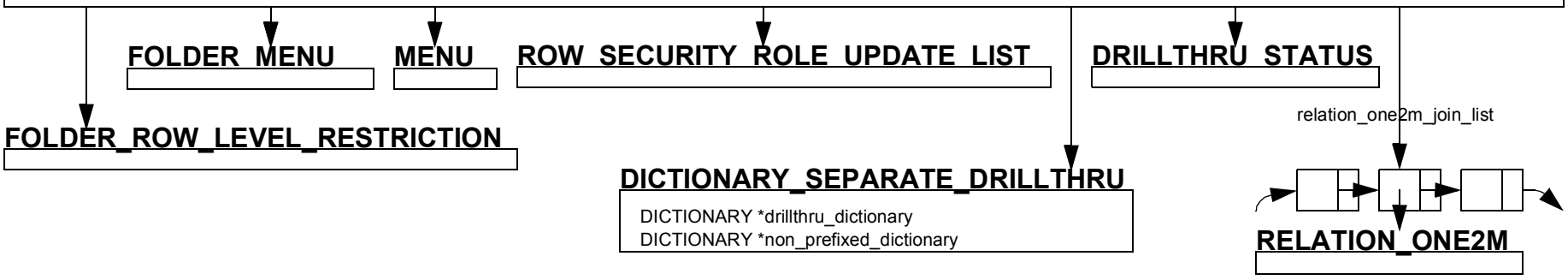
PROMPT_LOOKUP_INPUT (1)

```
/* Usage */
PROMPT_LOOKUP_INPUT *prompt_lookup_input_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    boolean application_menu_horizontal_boolean(),
    char *data_directory,
    DICTIONARY *original_post_dictionary );

/* Process */
PROMPT_LOOKUP_INPUT *prompt_lookup_input_calloc(
    void );

LIST *role_folder_list(
    role_name,
    folder_name );

boolean role_folder_lookup_boolean(
    APPASERVER_LOOKUP_STATE,
    APPASERVER_UPDATE_STATE,
    folder_name,
    role_folder_list );

boolean prompt_lookup_input_forbid(
    boolean role_folder_lookup_boolean() );

if ( prompt_lookup_input_forbid() ) return this;

boolean role_folder_insert_boolean =
    role_folder_state_boolean(
        APPASERVER_INSERT_STATE,
        folder_name,
        role_folder_list );

ROLE *role =
    role_fetch(
        role_name,
        1 /* fetch_role_attribute_exclude_list */ );

LIST *exclude_attribute_name_list = list_new();

list_set_list(
    exclude_attribute_name_list,
    LIST *role_attribute_exclude_name_list(
        APPASERVER_LOOKUP_STATE,
        role->role_attribute_exclude_list ) );

list_set_list(
    exclude_attribute_name_list,
    LIST *role_attribute_exclude_name_list(
        APPASERVER_UPDATE_STATE,
        role->role_attribute_exclude_list ) );

FOLDER *folder =
    folder_fetch(
        folder_name,
        exclude_attribute_name_list,
        /* Sets folder_attribute_primary_list */
        /* Sets folder_attribute_primary_key_list */
        /* Sets folder_attribute_name_list */
        1 /* fetch_folder_attribute_list */,
        1 /* fetch_attribute */,
        1 /* cache_boolean */ );

LIST *relation_mto1_list(
    folder_name
        /* many_folder_name */,
    folder->folder_attribute_primary_key_list
        /* many_folder_primary_key_list */ );

relation_mto1_list() =
    LIST *relation_mto1_without_omit_drillthru_list(
        relation_mto1_list() );

LIST *relation_mto1_isa_list(
    (LIST *)0 /* mto1_isa_list Pass in null */,
    folder_name
        /* many_folder_name */,
    folder->folder_attribute_primary_key_list
        /* many_folder_primary_key_list */,
    0 /* not fetch_relation_one2m_list */,
    1 /* fetch_relation_mto1_list */ );

LIST *folder_attribute_append_isa_list(
    folder->folder_attribute_list,
    relation_mto1_isa_list() );

LIST *folder_operation_list(
    folder_name,
    role_name,
    0 /* not fetch_operation */,
    0 /* not fetch_process */ );
```



PROMPT_LOOKUP_INPUT (2)

```
/* Process (continued) */
FOLDER_ROW_LEVEL_RESTRICTION *
    folder_row_level_restriction =
        folder_row_level_restriction_fetch(
            folder_name );

if ( application_menu_horizontal_boolean )
{
    FOLDER_MENU *folder_menu =
        folder_menu_fetch(
            application_name,
            session_key,
            role_name,
            data_directory,
            0 /* not folder_menu_remove_boolean */ );

    MENU *menu =
        menu_new(
            application_name,
            session_key,
            login_name,
            role_name,
            1 /* application_menu_horizontal_boolean */,
            folder_menu->count_list );
}

ROW_SECURITY_ROLE_UPDATE_LIST *
    row_security_role_update_list =
        /* Returns null if not participating */
        row_security_role_update_list_fetch(
            role_name,
            folder_name,
            role->override_row_restrictions );

LIST *folder_attribute_date_name_list(
    folder->folder_attribute_append_isa_list ),

DICTIONARY_SEPARATE_DRILLTHRU *
    dictionary_separate_drillthru =
        dictionary_separate_drillthru_new(
            original_post_dictionary,
            application_name,
            login_name,
            folder_attribute_date_name_list(),
            folder_attribute_append_isa_list() );

DRILLTHRU_STATUS *drillthru_status =
    drillthru_status_new(
        dictionary_separate_drillthru->
            drillthru_dictionary );

if ( drillthru_status_new()->skipped_boolean )
{
    relation_mto1_list =
        LIST *relation_mto1_foreign_key_less_equal_list(
            QUERY_DROP_DOWN_FETCH_MAX_ROWS,
            DRILLTHRU_SKIPPED_MAX_FOREIGN_LENGTH,
            relation_mto1_list() );
}

if (    !drillthru_status->participating_boolean
||      !drillthru_status->skipped_boolean )
{
    LIST *relation_one2m_join_list(
        folder_name
            /* one_folder_name */,
        folder->folder_attribute_primary_key_list
            /* one_folder_primary_key_list */,
        (DICTIONARY *)0 /* no_display_dictionary */ );
}

LIST *relation_one2m_pair_list(
    folder_name
        /* one_folder_name */,
    folder->folder_attribute_primary_key_list
        /* one_primary_key_list */ );

int relation_one2m_pair_list_length =
    list_length( relation_one2m_pair_list() );

/* Attributes */
LIST *role_folder_list;
boolean role_folder_lookup_boolean;
boolean forbid;
boolean role_folder_insert_boolean;
ROLE *role;
LIST *exclude_attribute_name_list;
FOLDER *folder;
LIST *relation_mto1_list;
LIST *relation_mto1_isa_list;
LIST *folder_attribute_append_isa_list;
LIST *folder_operation_list;
FOLDER_ROW_LEVEL_RESTRICTION *
    folder_row_level_restriction;
FOLDER_MENU *folder_menu;
MENU *menu;
ROW_SECURITY_ROLE_UPDATE_LIST *
    row_security_role_update_list;
DICTIONARY_SEPARATE_DRILLTHRU *
    dictionary_separate_drillthru;
LIST *folder_attribute_date_name_list;
DRILLTHRU_STATUS *drillthru_status;
LIST *relation_one2m_join_list;
int relation_one2m_pair_list_length;
```



POST_PROMPT_LOOKUP (1)

```
/* Usage */
POST_PROMPT_LOOKUP *post_prompt_lookup_new(
    int argc,
    char **argv,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    DICTIONARY *post_dictionary->
        original_post_dictionary );

/* Process */
POST_PROMPT_LOOKUP *post_prompt_lookup_calloc(
        void );

SESSION_FOLDER *session_folder =
    /* Sets appaserver environment and outputs argv */
    /* Each parameter is security inspected */
    session_folder_integrity_exit(
        argc,
        argv,
        application_name,
        session_key,
        login_name,
        role_name,
        folder_name,
        APPASERVER_LOOKUP_STATE,
        APPASERVER_UPDATE_STATE );

POST_PROMPT_LOOKUP_INPUT *
    post_prompt_lookup_input =
        post_prompt_lookup_input_new(
            FORM_RADIO_LIST_NAME,
            application_name,
            login_name,
            role_name,
            folder_name,
            original_post_dictionary );

if ( string_strcmp(
        post_prompt_lookup_input->selection,
        RADIO_LOOKUP_VALUE ) == 0 )
{
    QUERY_DROP_DOWN_LIST *
        query_drop_down_list_new(
            post_prompt_lookup_input->
                relation_mto1_list,
            post_prompt_lookup_input->
                dictionary_separate->
                prompt_dictionary /* dictionary */,
            (char *)0 /* many_table_name */ );

    QUERY_ATTRIBUTE_LIST *query_attribute_list_new(
        post_prompt_lookup_input->
            folder_attribute_append_isa_list,
        post_prompt_lookup_input->
            dictionary_separate->
            non_prefixed_dictionary /* dictionary */,
        list_length(
            post_prompt_lookup_input->
                relation_mto1_isa_list )
                /* relation_mto1_isa_list_length */,
        (char *)0
        /* row_security_role_update_list_where */ );

    LIST *post_prompt_lookup_include_name_list(
        query_drop_down_list_new(),
        query_attribute_list_new() );

    LIST *post_prompt_lookup_missing_required_list(
        LIST *post_prompt_lookup_input->
            folder_attribute_lookup_required_name_list,
        LIST *post_prompt_lookup_include_name_list() );

    if ( list_length(
            post_prompt_lookup_missing_required_list() ) )
    {
        char *post_prompt_lookup_missing_html(
            POST_PROMPT_LOOKUP_MISSING_MESSAGE,
            folder_name,
            post_prompt_lookup_missing_required_list() );
    }
}
```

SESSION_FOLDER

POST_PROMPT_LOOKUP_INPUT

QUERY_DROP_DOWN_LIST

QUERY_ATTRIBUTE_LIST

POST_PROMPT_LOOKUP (2)

```
/* Process (continued) */
DICTIONARY *dictionary_separate_send_dictionary(
    (DICTIONARY *)0 /* sort_dictionary */,
    DICTIONARY_SEPARATE_SORT_PREFIX,
    (DICTIONARY *)0 /* query_dictionary */,
    DICTIONARY_SEPARATE_QUERY_PREFIX,
    post_prompt_lookup_input->
        dictionary_separate->
        drillthru_dictionary,
    DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
    post_prompt_lookup_input->
        dictionary_separate->
        no_display_dictionary,
    DICTIONARY_SEPARATE_IGNORE_PREFIX,
    post_prompt_lookup_input->
        dictionary_separate->
        no_display_dictionary,
    DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
    (DICTIONARY *)0 /* pair_dictionary */,
    DICTIONARY_SEPARATE_PAIR_PREFIX,
    post_prompt_lookup_input->
        dictionary_separate->
        non_prefixed_dictionary );

char *dictionary_separate_send_string(
    dictionary_separate_send_dictionary() );

char *post_prompt_lookup_execute_system_string(
    application_name,
    session_key,
    login_name,
    role_name,
    folder_name,
    post_prompt_lookup_input->
        dictionary_separate->
        non_prefixed_dictionary,
    post_prompt_lookup_input->
        appaserver_error_filename,
    post_prompt_lookup_input->selection,
    dictionary_separate_send_string() );

/* Attributes */
SESSION_FOLDER *session_folder;
POST_PROMPT_LOOKUP_INPUT *
    post_prompt_lookup_input;
QUERY_DROP_DOWN_LIST *query_drop_down_list;
QUERY_ATTRIBUTE_LIST *query_attribute_list;
LIST *include_name_list;
LIST *missing_required_list;
char *missing_html;
DICTIONARY *dictionary_separate_send_dictionary;
char *dictionary_separate_send_string;
char *execute_system_string;
```

DICTIONARY

POST_PROMPT_LOOKUP (3)

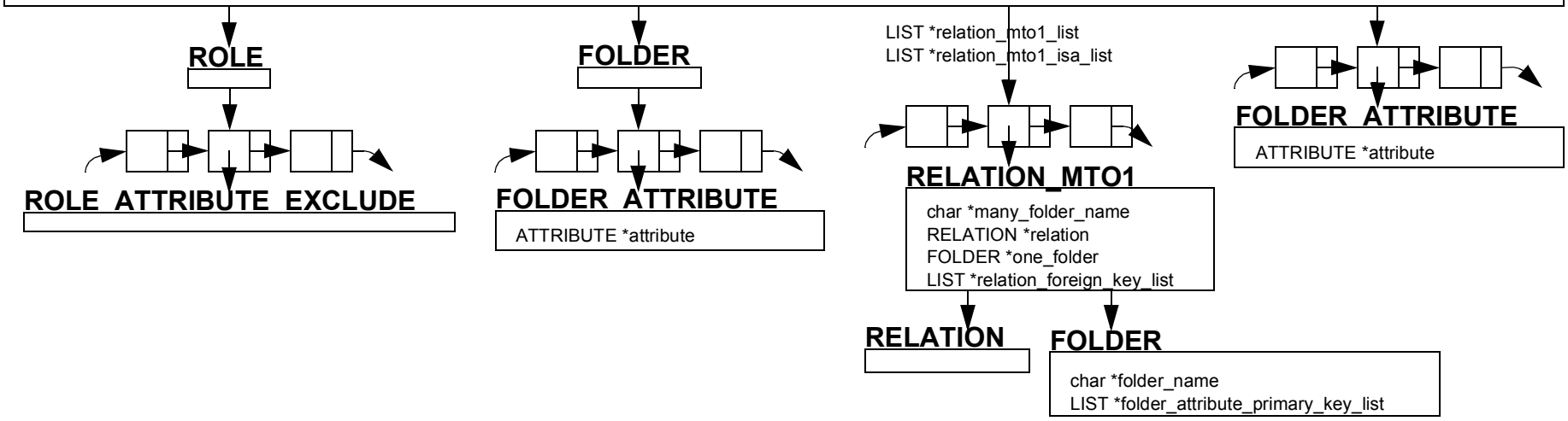
```
/* Usage */
char *post_prompt_lookup_execute_system_string(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    DICTIONARY *non_prefixed_dictionary,
    char *appaserver_error_filename(),
    char *post_prompt_lookup_input_selection(),
    char *dictionary_separate_send_string() );

/* Process */
if ( strcmp(
        post_prompt_lookup_input_selection,
        RADIO_STATISTICS_VALUE ) == 0 )
{
    char *execute_system_string =
        char *execute_system_string_lookup(
            LOOKUP_STATISTIC_EXECUTABLE,
            login_name,
            role_name,
            folder_name,
            dictionary_separate_send_string,
            appaserver_error_filename );
}
else
if ( strcmp(
        post_prompt_lookup_input_selection,
        RADIO_GROUP_COUNT_VALUE ) == 0 )
{
    char *execute_system_string =
        char *execute_system_string_lookup(
            GROUP_COUNT_EXECUTABLE,
            login_name,
            role_name,
            folder_name,
            dictionary_separate_send_string,
            appaserver_error_filename );
}
else
if ( strcmp(
        post_prompt_lookup_input_selection,
        RADIO_SPREADSHEET_VALUE ) == 0 )
{
    char *execute_system_string =
        char *execute_system_string_session(
            LOOKUP_SPREADSHEET_EXECUTABLE,
            session_key,
            login_name,
            role_name,
            folder_name,
            dictionary_separate_send_string,
            appaserver_error_filename );
}
else
if ( strcmp(
        post_prompt_lookup_input_selection,
        RADIO_INSERT_VALUE ) == 0 )
{
    char *execute_system_string =
        char *execute_system_string_post_prompt_insert(
            POST_PROMPT_INSERT_EXECUTABLE,
            application_name,
            session_key,
            login_name,
            role_name,
            folder_name,
            dictionary_separate_send_string,
            appaserver_error_filename );
}
else
if ( strcmp(
        post_prompt_lookup_input_selection,
        RADIO_DELETE_VALUE ) == 0 )
{
    char *execute_system_string =
        char *execute_system_string_lookup_delete(
            LOOKUP_DELETE_EXECUTABLE,
            application_name,
            session_key,
            login_name,
            role_name,
            folder_name,
            dictionary_separate_send_string,
            appaserver_error_filename(),
            "one" /* one_two_state */ );
}
else
if ( strcmp(
        post_prompt_lookup_input_selection,
        RADIO_GRACE_VALUE ) == 0 )
{
    char *execute_system_string =
        char *execute_system_string_session(
            LOOKUP_GRACE_EXECUTABLE,
            application_name,
            session_key,
            login_name,
            role_name,
            folder_name,
            dictionary_separate_send_string,
            appaserver_error_filename() );
}
else
if ( strcmp(
        post_prompt_lookup_input_selection,
        RADIO_GOOGLE_VALUE ) == 0 )
{
    char *execute_system_string =
        char *execute_system_string_session(
            LOOKUP_GOOGLE_EXECUTABLE,
            application_name,
            session_key,
            login_name,
            role_name,
            folder_name,
            dictionary_separate_send_string,
            appaserver_error_filename() );
}
else
if ( strcmp(
        post_prompt_lookup_input_selection,
        RADIO_HISTOGRAM_VALUE ) == 0 )
{
    char *execute_system_string =
        char *execute_system_string_session(
            LOOKUP_HISTOGRAM_EXECUTABLE,
            session_key,
            login_name,
            role_name,
            folder_name,
            dictionary_separate_send_string,
            appaserver_error_filename() );
}
else
```

POST_PROMPT_LOOKUP (4)

```
/* post_prompt_lookup_execute_system_string() */
if ( strcmp(
        post_prompt_lookup_input_selection,
        RADIO_SORT_VALUE ) == 0 )
{
    char *execute_system_string =
        char *execute_system_string_lookup_sort(
            LOOKUP_SORT_EXECUTABLE,
            application_name,
            session_key,
            login_name,
            role_name,
            folder_name,
            dictionary_separate_send_string,
            appaserver_error_filename() );
}
else
if ( strcmp(
        post_prompt_lookup_input_selection,
        RADIO_LOOKUP_VALUE ) == 0 )
{
    char *execute_system_string =
        char *execute_system_string_table_edit(
            TABLE_EDIT_EXECUTABLE,
            session_key,
            login_name,
            role_name,
            folder_name,
            FRAMESET_TABLE_FRAME /* target_frame */,
            (char *)0 /* results_string */,
            (char *)0 /* error_string */,
            dictionary_separate_send_string,
            appaserver_error_filename );
}
else
if ( strcmp(
        post_prompt_lookup_input_selection,
        RADIO_RCHART_VALUE ) == 0 )
{
    char *execute_system_string =
        char *execute_system_string_session(
            LOOKUP_RCHART_EXECUTABLE,
            session_key,
            login_name,
            role_name,
            folder_name,
            dictionary_separate_send_string,
            appaserver_error_filename() );
}

/* Usage */
LIST *post_prompt_lookup_include_name_list(
    QUERY_DROP_DOWN_LIST *query_drop_down_list,
    QUERY_ATTRIBUTE_LIST *query_attribute_list );

/* Process */
LIST *include_name_list = list_new();

list_set_list(
    include_name_list,
    query_drop_down_list->
        query_drop_down_relation_attribute_name_list );

list_set_list(
    include_name_list,
    LIST *query_attribute_list_name_list(
        query_attribute_list->
            query_date_time_between_list,
        query_attribute_list->list
            /* query_attribute_list */ ) );

/* Usage */
char *post_prompt_lookup_missing_html(
    char *POST_PROMPT_LOOKUP_MISSING_MESSAGE,
    char *folder_name,
    LIST *post_prompt_lookup_missing_required_list() );

/* Process */
char *prompt_lookup_title_string(
    APPASERVER_LOOKUP_STATE,
    folder_name );

sprintf(
    char missing_html[],
    "<h1>%s</h1>\n<h3>%s %s</h3>",
    prompt_lookup_title_string(),
    POST_PROMPT_LOOKUP_MISSING_MESSAGE,
    list_delimited(
        post_prompt_lookup_missing_required_list,
        ',' ) );

/* Driver */
if ( post_prompt_lookup->missing_html )
{
    void document_process_output(
        application_name,
        (LIST *)0 /* javascript_filename_list */,
        (char *)0 /* title_string */ );

    printf(   "%s\n",
              post_prompt_lookup->missing_html );

    void document_close();
}
else
{
    system(
        post_prompt_lookup->
            execute_system_string );
}
```

POST_PROMPT_LOOKUP_INPUT (1)

```
/* Usage */
POST_PROMPT_LOOKUP_INPUT *
   post_prompt_lookup_input_new(
      char *FORM_RADIO_LIST_NAME,
      char *application_name,
      char *login_name,
      char *role_name,
      char *folder_name,
      DICTIONARY *post_dictionary->
         original_post_dictionary );

/* Process */
POST_PROMPT_LOOKUP_INPUT *
   post_prompt_lookup_input_calloc(
      void );

ROLE *role =
   role_fetch(
      role_name,
      1 /* fetch_role_attribute_exclude_list */ );

LIST *role_attribute_exclude_lookup_name_list(
   APPASERVER_LOOKUP_STATE,
   APPASERVER_UPDATE_STATE,
   role->role_attribute_exclude_list );

FOLDER *folder =
   folder_fetch(
      folder_name,
      role_attribute_exclude_lookup_name_list(),
      /* Sets folder_attribute_primary_list */
      /* Sets folder_attribute_primary_key_list */
      /* Sets folder_attribute_name_list */
      1 /* fetch_folder_attribute_list */,
      1 /* fetch_attribute */,
      1 /* cache_boolean */ );

LIST *relation_mto1_list(
   folder_name
      /* many_folder_name */,
   folder->folder_attribute_primary_key_list
      /* many_folder_primary_key_list */ );

LIST *relation_mto1_isa_list(
   (LIST *)0 /* mto1_isa_list Pass in null */,
   folder_name
      /* many_folder_name */,
   folder->folder_attribute_primary_key_list
      /* many_folder_primary_key_list */,
   0 /* not fetch_relation_one2m_list */,
   0 /* not fetch_relation_mto1_list */ );

LIST *folder_attribute_append_isa_list(
   folder->folder_attribute_list
      /* append_isa_list */,
   relation_mto1_isa_list() );

LIST *folder_attribute_name_list(
   (char *)0 /* folder_name */,
   folder_attribute_append_isa_list() );

LIST *folder_attribute_date_name_list(
   folder_attribute_append_isa_list() );

LIST *folder_attribute_lookup_required_name_list(
   folder_attribute_append_isa_list() );
```



POST_PROMPT_LOOKUP_INPUT (2)

```
/* Process (continued) */
DICTIONARY_SEPARATE_POST_PROMPT_LOOKUP *
   dictionary_separate =
      dictionary_separate_post_prompt_lookup_new(
         original_post_dictionary,
         application_name,
         login_name,
         relation_mto1_list(),
         folder_attribute_name_list(),
         folder_attribute_date_name_list(),
         folder_attribute_append_isa_list() );

DRILLTHRU_STATUS *drillthru_status =
   drillthru_status_new(
      dictionary_separate->drillthru_dictionary );

char *post_prompt_lookup_input_selection(
   char *FORM_RADIO_LIST_NAME,
   DICTIONARY *dictionary_separate->
      non_prefixed_dictionary );

char *appaserver_error_filename(
   application_name );

/* Attributes */
ROLE *role;
LIST *role_attribute_exclude_lookup_name_list;
FOLDER *folder;
LIST *relation_mto1_list;
LIST *relation_mto1_isa_list;
LIST *folder_attribute_append_isa_list;
LIST *folder_attribute_name_list;
LIST *folder_attribute_date_name_list;
LIST *folder_attribute_lookup_required_name_list;
DICTIONARY_SEPARATE_POST_PROMPT_LOOKUP *
   dictionary_separate;
DRILLTHRU_STATUS *drillthru_status;
char *selection;
char *appaserver_error_filename;
```

DICTIONARY_SEPARATE_POST_PROMPT_LOOKUP

```
DICTIONARY *drillthru_dictionary
DICTIONARY *prompt_dictionary
DICTIONARY *no_display_dictionary
DICTIONARY *non_prefixed_dictionary
LIST *no_display_name_list
```

DRILLTHRU_STATUS

PROMPT_INSERT

```
/* Usage */
PROMPT_INSERT *prompt_insert_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    boolean application_menu_horizontal_boolean(),
    char *appaserver_parameter->data_directory,
    DICTIONARY *post_dictionary->
       original_post_dictionary );

/* Process */
PROMPT_INSERT *prompt_insert_calloc( void );

PROMPT_INSERT_INPUT *
    prompt_insert_input =
       prompt_insert_input_new(
          application_name,
          session_key,
          login_name,
          role_name,
          folder_name,
          application_menu_horizontal_boolean,
          data_directory,
          original_post_dictionary );

if ( prompt_insert_input->forbid ) exit( 1 );

if ( prompt_insert_input->
       drillthru_status->
       participating_boolean )
{
    char *post_choose_folder_action_string(
       POST_CHOOSE_FOLDER_EXECUTABLE,
       application_name,
       session_key,
       login_name,
       role_name,
       APPASERVER_INSERT_STATE,
       prompt_insert_input->
          drillthru_status->
          base_folder_name );
}
```

```
FORM_PROMPT_INSERT *form_prompt_insert =
    form_prompt_insert_new(
       PROMPT_INSERT_LOOKUP_CHECKBOX,
       application_name,
       session_key,
       login_name,
       role_name,
       folder_name,
       prompt_insert_input->
          folder->
          post_change_javascript,
       prompt_insert_input->
          folder_attribute_non_primary_name_list,
       prompt_insert_input->relation_mto1_list,
       prompt_insert_input->relation_mto1_isa_list,
       prompt_insert_input->
          folder_attribute_append_isa_list,
       prompt_insert_input->
          dictionary_separate_drillthru->
          drillthru_dictionary,
       prompt_insert_input->security_entity,
       prompt_insert_input->
          role_folder_lookup_boolean,
       prompt_insert_input->
          relation_one2m_pair_list,
       post_choose_folder_action_string() );

char *application_title_string( appliction_name );

char *prompt_insert_title_string(
    const char *APPASERVER_INSERT_STATE,
    char *folder_name );

int folder_attribute_calendar_date_name_list_length(
    prompt_insert_input->
       folder_attribute_append_isa_list );

char *document_head_calendar_setup_string(
    folder_attribute_calendar_date_name_list_length() );

LIST *prompt_insert_javascript_filename_list(
    char *prompt_insert_input->
       folder->
       javascript_filename );
```

```
DOCUMENT *document =
    document_new(
       application_name,
       application_title_string(),
       char *prompt_insert_title_string(),
       (char *)0 /* sub_title_html */,
       (char *)0 /* sub_sub_title_html */,
       prompt_insert_input->
          folder->
          notepad,
       (char *)0 /* onload_javascript_string */,
       (prompt_insert_input->menu)
          ? menu->html : (char *)0,
       document_head_menu_setup_string(
          (prompt_insert_input->menu)
             ? 1 : 0 /* menu_boolean */ ),
       document_head_calendar_setup_string(),
       javascript_include_string(
          application_name,
          prompt_insert_javascript_filename_list() ) );

char *document_form_html(
    document->html,
    document->document_head->html,
    document_head_close_tag(),
    document->document_body->html,
    form_prompt_insert->html,
    document_body_close_tag(),
    document_close_tag() );

free( form_prompt_insert->html );

/* Driver */
printf( "%s\n", prompt_insert->document_form_html );

/* Attributes */
PROMPT_INSERT_INPUT *prompt_insert_input;
char *post_choose_folder_action_string;
FORM_PROMPT_INSERT *form_prompt_insert;
char *application_title_string;
char *title_string;
int folder_attribute_calendar_date_name_list_length;
LIST *javascript_filename_list;
DOCUMENT *document;
char *document_form_html;
```

PROMPT_INSERT_INPUT

FORM_PROMPT_INSERT

DOCUMENT

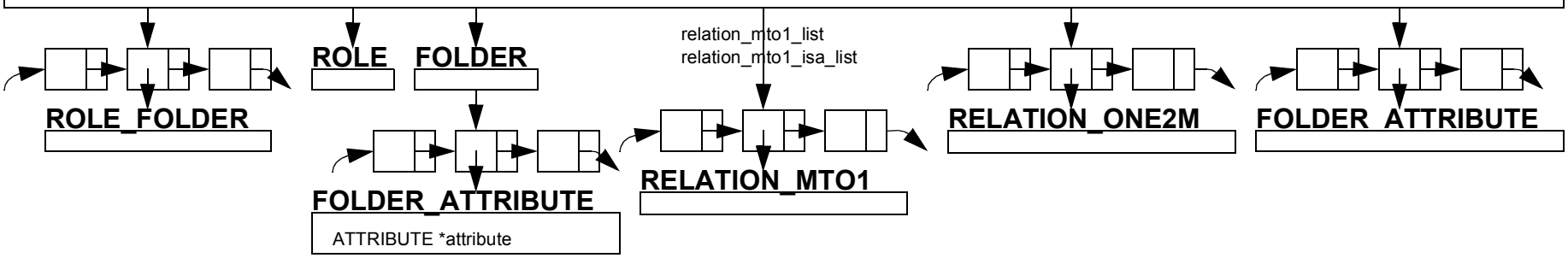
```
DOCUMENT_HEAD *document_head;
DOCUMENT_BODY *document_body;
```

PROMPT_INSERT_INPUT (1)

```
/* Usage */
PROMPT_INSERT_INPUT *prompt_insert_input_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    boolean application_menu_horizontal_boolean(),
    char *data_directory,
    DICTIONARY *original_post_dictionary );

/* Process */
PROMPT_INSERT_INPUT *
    prompt_insert_input_calloc(
        void );

LIST *role_folder_list(
    role_name,
    folder_name );

boolean role_folder_insert_boolean(
    APPASERVER_INSERT_STATE,
    folder_name,
    role_folder_list() );

boolean prompt_insert_input_forbid(
    boolean role_folder_insert_boolean() );

if ( prompt_insert_input_forbid() ) return this;
```

```
boolean role_folder_lookup_boolean(
    APPASERVER_LOOKUP_STATE,
    APPASERVER_UPDATE_STATE,
    folder_name,
    role_folder_list() );

ROLE *role =
    role_fetch(
        role_name,
        1 /* fetch_role_attribute_exclude_list */ );

LIST *role_attribute_exclude_insert_name_list(
    APPASERVER_INSERT_STATE
        /* role_permission_insert */,
    role->role_attribute_exclude_list );

FOLDER *folder =
    folder_fetch(
        folder_name,
        /* Set to cache all folders for a role */
        role_name,
        role_attribute_exclude_insert_name_list(),
        /* Sets folder_attribute_primary_list */
        /* Sets folder_attribute_primary_key_list */
        /* Sets folder_attribute_name_list */
        1 /* fetch_folder_attribute_list */,
        1 /* fetch_attribute */ );

LIST *folder_attribute_non_primary_name_list(
    folder->folder_attribute_list );
```

```
LIST *relation_mto1_list(
    folder_name
        /* many_folder_name */,
    folder->folder_attribute_primary_key_list
        /* many_primary_key_list */ );

LIST *relation_mto1_list() =
    relation_mto1_without_omit_drillthru_list(
        relation_mto1_list() );

LIST *relation_mto1_isa_list(
    (LIST *)0 /* mto1_isa_list Pass in null */,
    folder_name
        /* many_folder_name */,
    folder->folder_attribute_primary_key_list
        /* many_primary_key_list */,
    0 /* not fetch_relation_one2m_list */,
    0 /* not fetch_relation_mto1_list */ );

LIST *relation_one2m_pair_list(
    folder_name
        /* one_folder_name */,
    folder->folder_attribute_primary_key_list
        /* one_primary_key_list */ );

LIST *folder_attribute_append_isa_list(
    folder->folder_attribute_list /* append_isa_list */,
    relation_mto1_isa_list() );
```



PROMPT_INSERT_INPUT (2)

```
/* Process (continued) */
if ( application_menu_horizontal_boolean )
{
    FOLDER_MENU *folder_menu =
        folder_menu_fetch(
            application_name,
            session_key,
            role_name,
            data_directory,
            0 /* not folder_menu_remove_boolean */ );

    MENU *menu =
        menu_new(
            application_name,
            session_key,
            login_name,
            role_name,
            1 /* application_menu_horizontal_boolean */,
            folder_menu->count_list );
}

LIST *folder_attribute_date_name_list(
    folder_attribute_append_isa_list() );

DICTIONARY_SEPARATE_DRILLTHRU *
    dictionary_separate_drillthru =
        dictionary_separate_drillthru_new(
            original_post_dictionary,
            application_name,
            login_name,
            folder_attribute_date_name_list(),
            folder_attribute_append_isa_list() );

DRILLTHRU_STATUS *drillthru_status =
    drillthru_status_new(
        dictionary_separate_drillthru->
            drillthru_dictionary );

FOLDER_ROW_LEVEL_RESTRICTION *
    folder_row_level_restriction =
        folder_row_level_restriction_fetch(
            folder_name );

SECURITY_ENTITY *security_entity =
    security_entity_new(
        login_name,
        folder_name,
        folder_row_level_restriction->non_owner_forbid,
        role->override_row_restrictions );

/* Attributes */
LIST *role_folder_list;
boolean role_folder_insert_boolean;
boolean forbid;
boolean role_folder_lookup_boolean;
ROLE *role;
LIST *role_attribute_exclude_insert_name_list;
FOLDER *folder;
LIST *folder_attribute_non_primary_name_list;
LIST *relation_mto1_list;
LIST *relation_mto1_isa_list;
LIST *relation_one2m_pair_list;
LIST *folder_attribute_append_isa_list;
FOLDER_MENU *folder_menu;
MENU *menu;
LIST *folder_attribute_date_name_list;
DICTIONARY_SEPARATE_DRILLTHRU *
    dictionary_separate_drillthru;
DRILLTHRU_STATUS *drillthru_status;
FOLDER_ROW_LEVEL_RESTRICTION *
    folder_row_level_restriction;
SECURITY_ENTITY *security_entity;
```

FOLDER_MENU

MENU

DRILLTHRU_STATUS

SECURITY_ENTITY

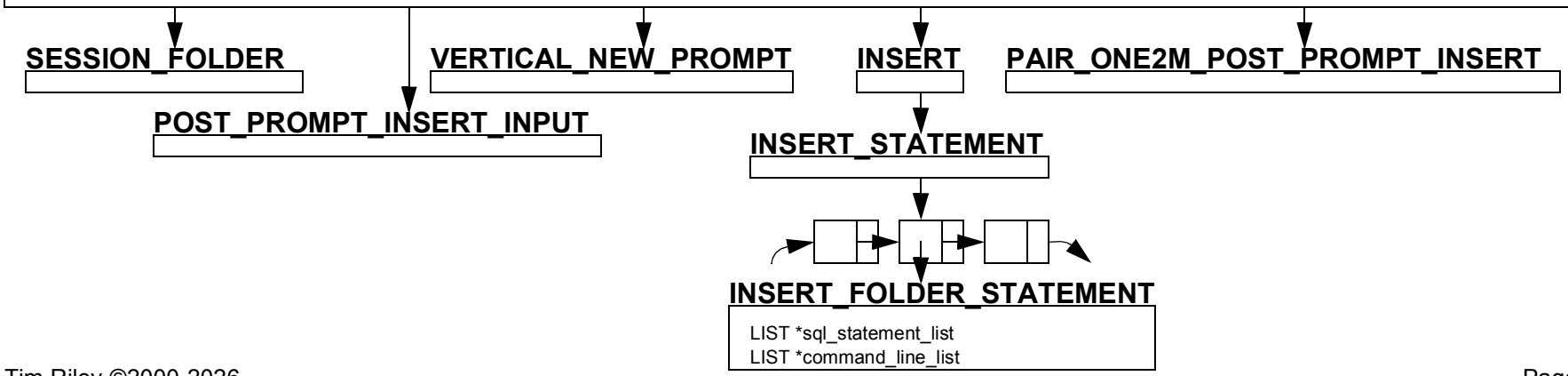
DICTIONARY_SEPARATE_DRILLTHRU

DICTIONARY *drillthru_dictionary
DICTIONARY *non_prefixed_dictionary

FOLDER_ROW_LEVEL_RESTRICTION

POST_PROMPT_INSERT (1)

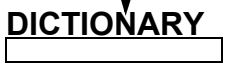
```
/* Usage */
POST_PROMPT_INSERT *post_prompt_insert_new(
    int argc,
    char **argv,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name );

/* Process */
POST_PROMPT_INSERT *
    post_prompt_insert_calloc(
        void );

SESSION_FOLDER *session_folder =
    /* Sets appaserver environment and outputs argv */
    /* Each parameter is security inspected */
    session_folder_integrity_exit(
        argc,
        argv,
        application_name,
        session_key,
        login_name,
        role_name,
        folder_name,
        APPASERVER_INSERT_STATE );

POST_PROMPT_INSERT_INPUT *
    post_prompt_insert_input =
        post_prompt_insert_input_new(
            application_name,
            session_key,
            login_name,
            role_name,
            folder_name );

VERTICAL_NEW_PROMPT *vertical_new_prompt_new(
    post_prompt_insert_input->
        dictionary_separate->
        non_prefixed_dictionary,
    VERTICAL_NEW_CHECKBOX_PREFIX,
    VERTICAL_NEW_PROMPT_ONE_HIDDEN_LABEL,
    VERTICAL_NEW_PROMPT_MANY_HIDDEN_LABEL,
    TABLE_INSERT_EXECUTABLE,
    FRAMESET_TABLE_FRAME,
    application_name,
    session_key,
    login_name,
    role_name,
    folder_name /* many_folder_name */,
    post_prompt_insert_input->
        dictionary_separate->
        drillthru_dictionary );

if ( vertical_new_prompt_new() ) return this;

if ( !post_prompt_insert_input->lookup_boolean )
{
    INSERT *insert_new(
        application_name,
        session_key,
        login_name,
        role_name,
        folder_name,
        post_prompt_insert_input->
            folder->
            folder_attribute_primary_key_list
            /* root_primary_key_list */,
        post_prompt_Insert_input->
            folder_attribute_append_isa_list,
        post_prompt_insert_input->
            relation_mto1_list,
        post_prompt_insert_input->
            relation_mto1_isa_list,
        post_prompt_insert_input->
            dictionary_separate->
            prompt_dictionary
            /* in/out */,
        (DICTIONARY *)0 /* multi_row_dictionary */,
        post_prompt_insert_input->
            dictionary_separate->
            ignore_name_list,
        post_prompt_insert_input->
            post_change_process );

    PAIR_ONE2M_POST_PROMPT_INSERT *
        pair_one2m_post_prompt_insert =
            pair_one2m_post_prompt_insert_new(
                PAIR_ONE2M_ONE_FOLDER_KEY,
                PAIR_ONE2M_MANY_FOLDER_KEY,
                PAIR_ONE2M_UNFULFILLED_LIST_KEY,
                PAIR_ONE2M_FOLDER_DELIMITER,
                post_prompt_insert_input->
                    dictionary_separate->
                    pair_dictionary );
}
```



POST_PROMPT_INSERT (2)

```
/* Process (continued) */
if (    pair_one2m_post_prompt_insert
&&   pair_one2m_post_prompt_insert->
         one_folder_name
&&   !insert )
{
   LIST *post_prompt_insert_missing_primary_key_list(
      post_prompt_insert_input->
         folder->
         folder_attribute_primary_key_list,
      post_prompt_insert_input->
         dictionary_separate->
         prompt_dictionary );

   char *post_prompt_insert_missing_display(
      POST_PROMPT_INSERT_MISSING_TEMPLATE,
      post_prompt_insert_missing_primary_key_list() );

   return this;
}

DICTIONARY *post_prompt_insert_query_dictionary(
      QUERY_RELATION_OPERATOR_PREFIX,
      QUERY_EQUAL,
      post_prompt_insert_input->
         dictionary_separate->
         prompt_dictionary );

DICTIONARY *dictionary_separate_send_dictionary(
   (DICTIONARY *)0 /* sort_dictionary */,
   DICTIONARY_SEPARATE_SORT_PREFIX,
   post_prompt_insert_query_dictionary(),
   DICTIONARY_SEPARATE_QUERY_PREFIX,
   post_prompt_insert_input->
      dictionary_separate->
      drillthru_dictionary,
   DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
   post_prompt_insert_input->
      dictionary_separate->
      ignore_dictionary,
   DICTIONARY_SEPARATE_IGNORE_PREFIX,
   post_prompt_insert_input->
      dictionary_separate->
      ignore_dictionary
         /* no_display_dictionary if lookup button */,
   DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
   post_prompt_insert_input->
      dictionary_separate->
      pair_dictionary,
   DICTIONARY_SEPARATE_PAIR_PREFIX,
   post_prompt_insert_input->
      dictionary_separate->
      prompt_dictionary
         /* non_prefixed_dictionary */ );

char *dictionary_separate_send_string(
   dictionary_separate_send_dictionary() );

/* Attributes */
SESSION_FOLDER *session_folder;
POST_PROMPT_INSERT_INPUT *
   post_prompt_insert_input;
VERTICAL_NEW_PROMPT *vertical_new_prompt;
INSERT *insert;
PAIR_ONE2M_POST_PROMPT_INSERT *
   pair_one2m_post_prompt_insert;
LIST *missing_primary_key_list;
char *missing_display;
DICTIONARY *query_dictionary;
DICTIONARY *dictionary_separate_send_dictionary;
char *dictionary_separate_send_string;
```

query_dictionary
dictionary_separate_send_dictionary



POST_PROMPT_INSERT (3)

```
/* Usage */
DICTIONARY *post_prompt_insert_query_dictionary(
    const char *QUERY_RELATION_OPERATOR_PREFIX,
    const char *QUERY_EQUAL,
    DICTIONARY *prompt_dictionary );

/* Process */
DICTIONARY *query_prompt_query_dictionary(
    QUERY_RELATION_OPERATOR_PREFIX,
    QUERY_EQUAL,
    prompt_dictionary );

/* Usage */
LIST *post_prompt_insert_missing_primary_key_list(
    LIST *folder_attribute_primary_key_list,
    DICTIONARY *prompt_dictionary );

/* Process */
LIST *missing_primary_key_list = list_new();

for char *attribute_name in
    folder_attribute_primary_key_list
{
    INSERT_DATUM *insert_datum_extract(
        prompt_dictionary,
        (DICTIONARY *)0 /* multi_row_dictionary */,
        0 /* row_number */,
        attribute_name,
        1 /* primary_key_index */,
        0 /* attribute_is_number */ );

    if ( !insert_datum_extract() )
    {
        list_set(
            missing_primary_key_list,
            attribute_name );
}

/* Usage */
char *post_prompt_insert_missing_display(
    char *POST_PROMPT_INSERT_MISSING_TEMPLATE,
    LIST *post_prompt_insert_missing_primary_key_list )

/* Process */
snprintf(
    display[],
    sizeof ( display ),
    POST_PROMPT_INSERT_MISSING_TEMPLATE,
    string_initial_capital(
        buffer[],
        list_display_delimited_plus_space(
            post_prompt_insert_missing_primary_key_list,
            ',' ) ) );

/* Usage */
boolean post_prompt_insert_fatal_duplicate_boolean(
    const char *CREATE_TABLE_UNIQUE_SUFFIX,
    const char *CREATE_TABLE_ADDITIONAL_SUFFIX,
    char *appaserver_table_name,
    char *insert_statement_error_string );

/* Process */
if ( !insert_statement_error_string ) return 0;

char *create_table_unique_index_name(
    CREATE_TABLE_UNIQUE_SUFFIX,
    appaserver_table_name );

boolean string_exists(
    insert_statement_error_string,
    create_table_unique_index_name() /* substring */ );

if ( string_exists() ) return 0;

char *create_table_unique_index_name(
    (char *)0 /* CREATE_TABLE_UNIQUE_SUFFIX */,
    appaserver_table_name );

boolean string_exists(
    insert_statement_error_string,
    create_table_unique_index_name() /* substring */ );

if ( string_exists() ) return 0;

char *create_table_additional_unique_index_name(
    CREATE_TABLE_ADDITIONAL_SUFFIX,
    appaserver_table_name );

boolean string_exists(
    insert_statement_error_string,
    create_table_additional_unique_index_name() );

if ( string_exists() ) return 0;

return 1;
```

POST_PROMPT_INSERT (4)

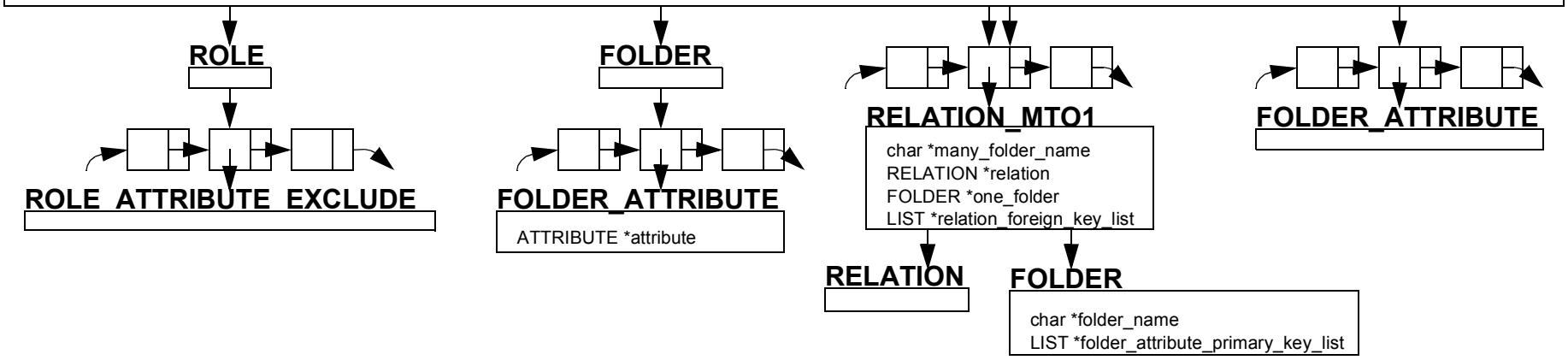
```
/* Driver */
if ( post_prompt_insert->vertical_new_prompt )
{
    system_string =
        post_prompt_insert->
            vertical_new_prompt->
            execute_system_string_table_insert;

    goto execute_system_string;
}

/* If pair one to many but left out a primary key */
if ( post_prompt_insert->missing_display )
{
    void document_process_output(
        application_name,
        (LIST *)0 /* javascript_filename_list */,
        "Insert Results" /* title_string */ );

    printf(
        "%s\n",
        post_prompt_insert->missing_display );

    document_close();
    exit( 0 );
}

/* If lookup button */
if (    post_prompt_insert->
            post_prompt_insert_input->
            lookup_boolean )
{
    char *system_string =
        execute_system_string_table_edit(
            TABLE_EDIT_EXECUTABLE,
            session_key,
            login_name,
            role_name,
            folder_name,
            FRAMESET_TABLE_FRAME /* target_frame */,
            results_string,
            insert_statement_error_string,
            post_prompt_insert->
                dictionary_separate_send_string,
            post_prompt_insert->
                post_prompt_insert_input->
                appaserver_error_filename );

    goto execute_system_string;
}

/* If included all the primary keys */
if ( post_prompt_insert->insert )
{
    char *insert_statement_error_string =
        insert_statement_execute(
            post_prompt_insert->
                insert->
                insert_statement->
                insert_folder_statement_list,
            post_prompt_insert->
                insert->
                appaserver_error_filespecification );

    char *appaserver_table_name(
        post_prompt_insert->
            post_prompt_insert_input->
            folder->
            folder_name );

    boolean
     post_prompt_insert_fatal_duplicate_boolean(
        CREATE_TABLE_UNIQUE_SUFFIX,
        CREATE_TABLE_ADDITIONAL_SUFFIX,
        appaserver_table_name(),
        insert_statement_error_string );

    if ( post_prompt_insert_fatal_duplicate_boolean() )
    {
        void document_process_output(
            application_name,
            (LIST *)0 /* javascript_filename_list */,
            "Duplicate Error" /* title_string */ );

        printf(
            "<h3>%s</h3>\n",
            insert_statement_error_string );

        document_close();
        exit( 0 );
    }
    else
    {
        char *result_string =
                post_prompt_insert->
                insert->
                results_string;
    }
}

#ifdef NOT_DEFINED
/* If pair one to many is finished */
if (    post_prompt_insert->
            pair_one2m_post_prompt_insert->
            one_folder_name
&&  post_prompt_insert->
            pair_one2m_post_prompt_insert->
            many_folder_name
&&  !post_prompt_insert->
            pair_one2m_post_prompt_insert->
            next_folder_name )
{
    document_process_output(
        application_name,
        (LIST *)0 /* javascript_filename_list */,
        "Insert Complete" /* title_string */ );

    document_close();

    exit( 0 );
}
#endif
```

POST_PROMPT_INSERT (5)

```
/* Driver (continued) */

/* If pair one to many is skipped */
if (    post_prompt_insert->
            pair_one2m_post_prompt_insert->
                one_folder_name
&&   !post_prompt_insert->
            pair_one2m_post_prompt_insert->
                many_folder_name )
{
    char *system_string =
        execute_system_string_table_edit(
            TABLE_EDIT_EXECUTABLE,
            session_key,
            login_name,
            role_name,
            folder_name,
            FRAMESET_TABLE_FRAME /* target_frame */,
            result_string,
            insert_statement_error_string,
            post_prompt_insert->
                dictionary_separate_send_string,
            post_prompt_insert->
                post_prompt_insert_input->
                appaserver_error_filespecification );

    goto execute_system_string;
}

/* If pair one to many has a next */
if (    post_prompt_insert->
            pair_one2m_post_prompt_insert->
            next_folder_name
&&   *post_prompt_insert->
            pair_one2m_post_prompt_insert->
            next_folder_name )
{
    char *system_string =
        execute_system_string_table_insert(
            TABLE_INSERT_EXECUTABLE,
            session_key,
            login_name,
            role_name,
            post_prompt_insert->
                pair_one2m_post_prompt_insert->
                next_folder_name,
            FRAMESET_TABLE_FRAME /* target_frame */,
            result_string,
            insert_statement_error_string,
            post_prompt_insert->
                dictionary_separate_send_string,
            post_prompt_insert->
                post_prompt_insert_input->
                appaserver_error_filespecification );

    goto execute_system_string;
}

/* If left out a primary key */
if ( !post_prompt_insert->insert )
{
    char *system_string =
        execute_system_string_table_insert(
            TABLE_INSERT_EXECUTABLE,
            session_key,
            login_name,
            role_name,
            folder_name,
            FRAMESET_TABLE_FRAME /* target_frame */,
            result_string,
            insert_statement_error_string,
            post_prompt_insert->
                dictionary_separate_send_string,
            post_prompt_insert->
                post_prompt_insert_input->
                appaserver_error_filespecification );

    goto execute_system_string;
}

char *system_string =
    execute_system_string_table_edit(
        TABLE_EDIT_EXECUTABLE,
        session_key,
        login_name,
        role_name,
        folder_name,
        FRAMESET_TABLE_FRAME /* target_frame */,
        result_string,
        insert_statement_error_string,
        post_prompt_insert->
            dictionary_separate_send_string,
        post_prompt_insert->
            post_prompt_insert_input->
            appaserver_error_filespecification );

execute_system_string:
system( system_string );
```

POST_PROMPT_INSERT_INPUT (1)

```
/* Usage */
POST_PROMPT_INSERT_INPUT *
   post_prompt_insert_input_new(
      char *application_name,
      char *session_key,
      char *login_name,
      char *role_name,
      char *folder_name );

/* Process */
POST_PROMPT_INSERT_INPUT *
   post_prompt_insert_input_calloc(
      void );

ROLE *role =
   role_fetch(
      role_name,
      1 /* fetch_role_attribute_exclude_list */ );

LIST *role_attribute_exclude_insert_name_list(
   APPASERVER_INSERT_STATE,
   role->role_attribute_exclude_list );

FOLDER *folder =
   folder_fetch(
      folder_name,
      role_attribute_exclude_insert_name_list(),
      /* Sets folder_attribute_primary_list */
      /* Sets folder_attribute_primary_key_list */
      /* Sets folder_attribute_name_list */
      1 /* fetch_folder_attribute_list */,
      1 /* fetch_attribute */,
      1 /* cache_boolean */ );

LIST *relation_mto1_list(
   folder_name
      /* many_folder_name */,
   folder->folder_attribute_primary_key_list
      /* many_folder_primary_key_list */ );

LIST *relation_mto1_isa_list(
   (LIST *)0 /* mto1_isa_list Pass in null */,
   folder_name
      /* many_folder_name */,
   folder->folder_attribute_primary_key_list
      /* many_primary_key_list */,
   0 /* not fetch_relation_one2m_list */,
   0 /* not fetch_relation_mto1_list */ );

LIST *folder_attribute_append_isa_list(
   folder->folder_attribute_list /* append_isa_list */,
   relation_mto1_isa_list() );
```



POST_PROMPT_INSERT_INPUT (2)

```
/* Process (continued) */
APPASERVER_PARAMETER *
   appaserver_parameter_new();

LIST *folder_attribute_upload_filename_list(
   folder_attribute_append_isa_list() );

POST_DICTIONARY *post_dictionary =
   post_dictionary_stdin_new(
      /* Used when expecting a spooled file */
      WIDGET_UPLOAD_RECALL_PREFIX,
      application_name,
      appaserver_parameter_new()->upload_directory,
      folder_attribute_upload_filename_list() );

LIST *folder_attribute_name_list(
   (char *)0 /* folder_name */,
   folder_attribute_append_isa_list() );

LIST *folder_attribute_date_name_list(
   folder_attribute_append_isa_list() );

DICTIONARY_SEPARATE_POST_PROMPT_INSERT *
   dictionary_separate =
      dictionary_separate_post_prompt_insert_new(
         post_dictionary->original_post_dictionary,
         application_name,
         login_name,
         relation_mto1_list(),
         folder_attribute_append_isa_list(),
         folder_attribute_name_list(),
         folder_attribute_date_name_list() );

boolean post_prompt_insert_input_lookup_boolean(
   char *PROMPT_INSERT_LOOKUP_CHECKBOX,
   DICTIONARY *dictionary_separate->
      non_prefixed_dictionary );

if ( !post_prompt_insert_input_lookup_boolean() )
{
   if ( folder->post_change_process_name )
   {
      PROCESS *post_change_process =
         process_fetch(
            folder->post_change_process_name,
            /* Not fetching javascript */
            (char *)0 /* document_root */,
            (char *)0 /* relative_source_directory */,
            1 /* check_executable_inside_filesystem */,
            appaserver_parameter_mount_point() );
   }

   APPASERVER_USER *insert_appaserver_user(
      APPASERVER_USER_FULL_NAME_COLUMN,
      login_name,
      role_attribute_exclude_insert_name_list(),
      folder_attribute_append_isa_list() );

   if ( insert_appaserver_user() )
   {
      dictionary_set(
         dictionary_separate->prompt_dictionary,
         APPASERVER_USER_FULL_NAME_COLUMN,
         insert_appaserver_user()->
            appaserver_user_full_name );

      dictionary_set(
         dictionary_separate->prompt_dictionary,
         APPASERVER_USER_CONTACT_KEY_COLUMN,
         insert_appaserver_user()->
            appaserver_user_contact_key );
   }
}

char *appaserver_error_filespecification(
   application_name );

/* Attributes */
ROLE *role;
LIST *role_attribute_exclude_insert_name_list;
FOLDER *folder;
LIST *relation_mto1_list;
LIST *relation_mto1_isa_list;
LIST *folder_attribute_append_isa_list;
APPASERVER_PARAMETER *appaserver_parameter;
LIST *folder_attribute_upload_filename_list;
POST_DICTIONARY *post_dictionary;
LIST *folder_attribute_name_list;
LIST *folder_attribute_date_name_list;
DICTIONARY_SEPARATE_POST_PROMPT_INSERT *
   dictionary_separate;
boolean lookup_boolean;
PROCESS *post_change_process;
APPASERVER_USER *insert_appaserver_user;
char *appaserver_error_filespecification;
```

POST_DICTIONARY

APPASERVER_PARAMETER

PROCESS

APPASERVER_USER

DICTIONARY_SEPARATE_POST_PROMPT_INSERT

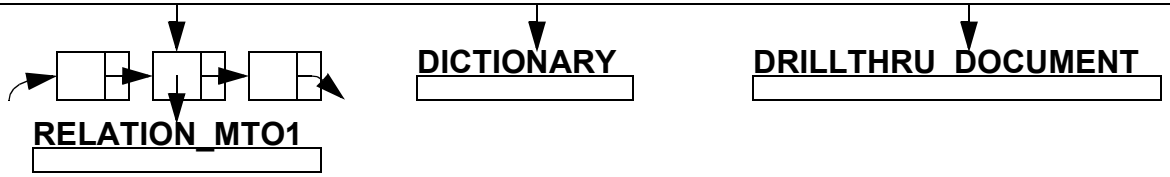
```
DICTIONARY *drillthru_dictionary
DICTIONARY *prompt_dictionary
DICTIONARY *pair_dictionary
DICTIONARY *ignore_dictionary
DICTIONARY *non_prefixed_dictionary
LIST *ignore_name_list
```

DRILLTHRU

```
/* Usage */
char *drillthru_current_folder_name(
    LIST *relation_mto1_drillthru_list() );

/* Process */
RELATION_MTO1 *relation_mto1 =
    list_last(
        relation_mto1_drillthru_list );

if ( relation_mto1 )
    return relation_mto1->one_folder_name;
else
    return (char *)0;
```

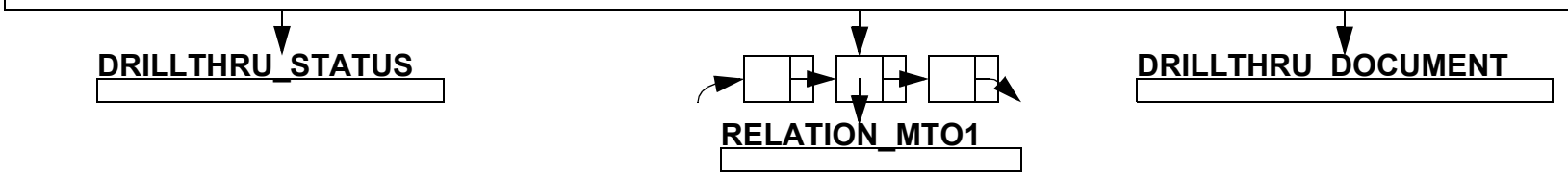
DRILLTHRU_START

```
/* Usage */
DRILLTHRU_START *drillthru_start_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *state,
    char *appaserver_parameter->
        appaserver_data_directory );

/* Process */
DRILLTHRU_START *drillthru_start_calloc(
    void );

LIST *folder_attribute_fetch_primary_key_list(
    folder_name );

LIST *relation_mto1_drillthru_list(
    (LIST *)0 /* relation_mto1_list Pass in null */,
    folder_name
        /* base_folder_name */,
    folder_attribute_fetch_primary_key_list()
        /* base_folder_primary_key_list */,
    (LIST *)0 /* fulfilled_folder_name_list */,
    1 /* cache_boolean */ );

boolean drillthru_start_participating_boolean(
    int length( relation_mto1_drillthru_list )
        /* relation_mto1_drillthru_list_length */ );

if ( !drillthru_start_participating_boolean() )
        return this;

DICTIONARY *drillthru_start_dictionary(
    const char *DRILLTHRU_BASE_KEY,
    char *folder_name );

char *drillthru_current_folder_name(
    relation_mto1_drillthru_list() );

DRILLTHRU_DOCUMENT *drillthru_document =
    drillthru_document_new(
        application_name,
        session_key,
        login_name,
        role_name,
        state,
        drillthru_dictionary,
        appaserver_data_directory,
        folder_name /* base_folder_name */,
        drillthru_current_folder_name(),
        1 /* first_time_boolean */ );

/* Attributes */
LIST *folder_attribute_fetch_primary_key_list;
LIST *relation_mto1_drillthru_list;
boolean participating_boolean;
DICTIONARY *drillthru_dictionary;
char *drillthru_current_folder_name;
DRILLTHRU_DOCUMENT *drillthru_document;
```



DRILLTHRU_CONTINUE

```
/* Usage */
DRILLTHRU_CONTINUE *drillthru_continue_new(
    DICTIONARY *drillthru_dictionary /* in/out */,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *current_folder_name,
    char *state,
    char *appaserver_parameter->
        appaserver_data_directory );

/* Process */
DRILLTHRU_CONTINUE *drillthru_continue_calloc(
    void );

DRILLTHRU_STATUS *drillthru_status_new(
      drillthru_dictionary );

if ( !drillthru_status->participating_boolean )
        return this;

if (  drillthru_status->skipped_boolean )
{
      if ( strcmp(
            state,
            APPASERVER_INSERT_STATE ) == 0 )
      {
        DRILLTHRU_DOCUMENT *drillthru_document =
            drillthru_document_new(
                application_name,
                session_key,
                login_name,
                role_name,
                state,
                drillthru_dictionary,
                appaserver_data_directory,
                drillthru_status->base_folder_name,
                current_folder_name,
                1 /* first_time_boolean */ );
      }
      return this;
}

LIST *drillthru_continue_fulfilled_folder_name_list(
    const char *DRILLTHRU_FULFILLED_KEY,
    const char DRILLTHRU_KEY_DELIMITER,
    DICTIONARY *drillthru_dictionary );

list_set(
    drillthru_continue_fulfilled_folder_name_list(),
    current_folder_name );

LIST *folder_attribute_fetch_primary_key_list(
    drillthru_status->base_folder_name );

LIST *relation_mto1_drillthru_list(
    (LIST *)0 /* relation_mto1_list Pass in null */,
    drillthru_status->base_folder_name,
    folder_attribute_fetch_primary_key_list()
        /* base_folder_primary_key_list */,
    drillthru_continue_fulfilled_folder_name_list() );

boolean drillthru_continue_completed_boolean(
    int list_length( relation_mto1_drillthru_list() )
        /* relation_mto1_drillthru_list_length */ );

if ( drillthru_continue_completed_boolean() )
        return this;

char *drillthru_current_folder_name(
    relation_mto1_drillthru_list() );

char *drillthru_continue_data_list_string(
    const char DRILLTHRU_KEY_DELIMITER,
    LIST *drillthru_continue_fulfilled_folder_name_list() );

void dictionary_set(
    drillthru_dictionary,
    DRILLTHRU_FULFILLED_KEY,
    drillthru_continue_data_list_string() );

dictionary_append_dictionary(
    drillthru_dictionary,
    query_dictionary );

DRILLTHRU_DOCUMENT *drillthru_document =
    drillthru_document_new(
        application_name,
        session_key,
        login_name,
        role_name,
        state,
        drillthru_dictionary,
        appaserver_data_directory,
        drillthru_status->base_folder_name,
        drillthru_current_folder_name(),
        0 /* not first_time_boolean */ );

/* Attributes */
DRILLTHRU_STATUS *drillthru_status;
LIST *fulfilled_folder_name_list;
LIST *folder_attribute_fetch_primary_key_list;
LIST *relation_mto1_drillthru_list;
boolean completed_boolean;
char *drillthru_current_folder_name;
char *data_list_string;
DRILLTHRU_DOCUMENT *drillthru_document;
```



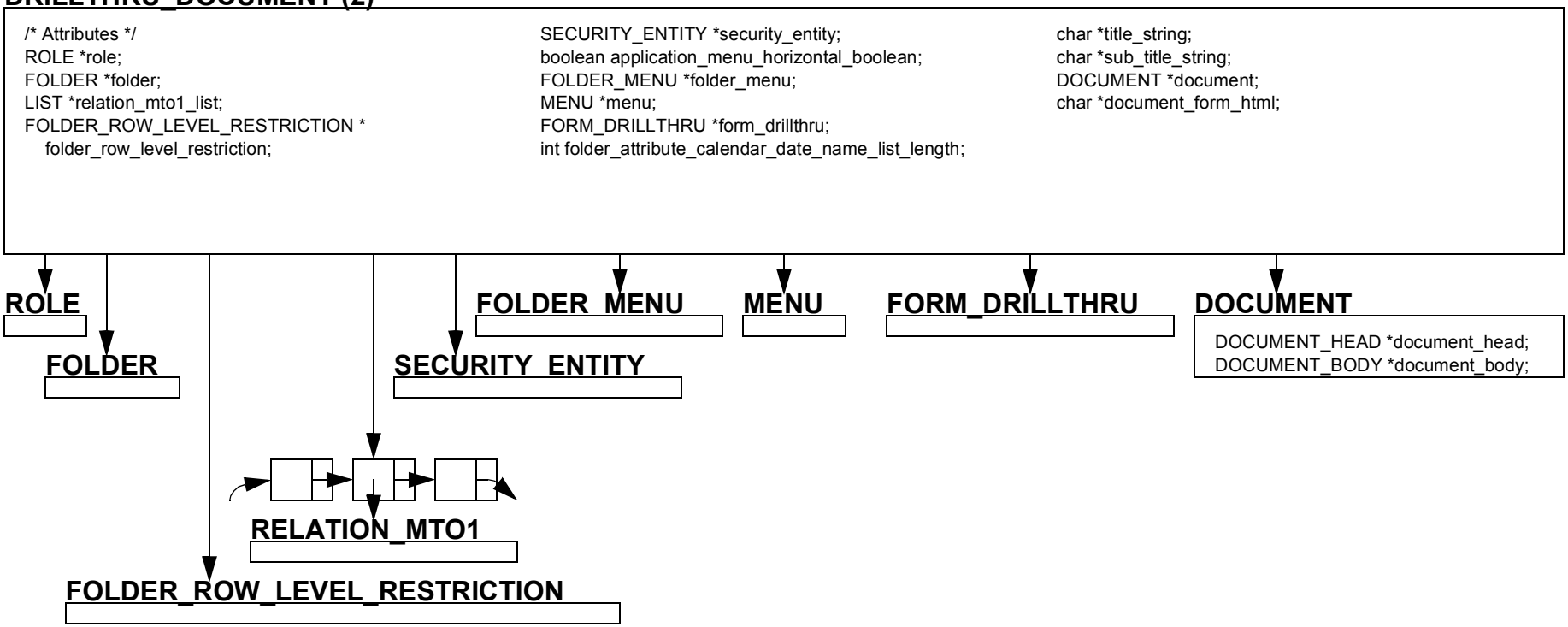
DRILLTHRU_STATUS

```
/* Usage */
DRILLTHRU_STATUS *drillthru_status_new(
     DICTIONARY *drillthru_dictionary );

/* Process */
DRILLTHRU_STATUS *drillthru_status_calloc(
     void );

char *drillthru_status_base_folder_name(
   const char *DRILLTHRU_BASE_KEY,
   DICTIONARY *drillthru_dictionary );

boolean drillthru_status_participating_boolean(
   char *drillthru_status_base_folder_name() );

boolean drillthru_status_first_time_boolean(
   const char *DRILLTHRU_FULFILLED_KEY,
   DICTIONARY *drillthru_dictionary );

boolean drillthru_status_skipped_boolean(
   DRILLTHRU_SKIPPED_KEY,
   drillthru_dictionary,
   drillthru_status_first_time_boolean() );

boolean drillthru_status_active_boolean(
   boolean drillthru_status_skipped_boolean() );

/* Usage */
boolean drillthru_status_skipped_boolean(
   const char *DRILLTHRU_SKIPPED_KEY,
   DICTIONARY *drillthru_dictionary,
   boolean drillthru_status_first_time_boolean );

/* Process */
int dictionary_length( drillthru_dictionary );

if ( !dictionary_length() ) return 0;

boolean dictionary_exists_key(
   drillthru_dictionary,
   DRILLTHRU_SKIPPED_KEY );

if ( dictionary_exists_key() )
{
   return 1;
}
else
if ( !drillthru_status_first_time_boolean )
   return 0;
else
if ( dictionary_length() > 1 )
   return 0;
else
   return 1;

/* Attributes */
char *base_folder_name;
boolean participating_boolean;
boolean first_time_boolean;
boolean skipped_boolean;
boolean active_boolean;
```

DRILLTHRU_DOCUMENT (1)

```
/* Usage */
DRILLTHRU_DOCUMENT *drillthru_document_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *state,
    DICTIONARY *drillthru_dictionary,
    char *appaserver_data_directory,
    char *base_folder_name,
    char *drillthru_current_folder_name(),
    boolean first_time_boolean );

/* Process */
DRILLTHRU_DOCUMENT *
    drillthru_document_calloc(
        void );

ROLE *role =
    role_fetch(
        role_name,
        0 /* not fetch_role_attribute_exclude_list */ );

FOLDER *folder =
    folder_fetch(
        drillthru_current_folder_name,
        /* Set to cache all folders for a role */
        role_name,
        (LIST *)0 /* exclude_attribute_name_list */,
        /* Sets folder_attribute_primary_list */
        /* Sets folder_attribute_primary_key_list */
        /* Sets folder_attribute_name_list */
        1 /* fetch_folder_attribute_list */,
        1 /* fetch_attribute */,
        1 /* cache_boolean */ );

LIST *relation_mto1_list(
    drillthru_current_folder_name,
    folder->folder_attribute_primary_key_list );

FOLDER_ROW_LEVEL_RESTRICTION *
    folder_row_level_restriction =
        folder_row_level_restriction_fetch(
            drillthru_current_folder_name );

SECURITY_ENTITY *security_entity =
    security_entity_new(
        login_name,
        base_folder_name,
        folder_row_level_restriction->
            non_owner_forbid,
        role->override_row_restrictions );

boolean application_menu_horizontal_boolean(
    application_name );

if ( application_menu_horizontal_boolean() )
{
    FOLDER_MENU *folder_menu =
        folder_menu_fetch(
            application_name,
            session_key,
            role_name,
            appaserver_data_directory,
            0 /* not folder_menu_remove_boolean */ );

    MENU *menu =
        menu_new(
            application_name,
            session_key,
            login_name,
            role_name,
            1 /* application_menu_horizontal_boolean */,
            folder_menu->count_list );
}

FORM_DRILLTHRU *form_drillthru =
    form_drillthru_new(
        application_name,
        session_key,
        login_name,
        role_name,
        state,
        base_folder_name,
        drillthru_current_folder_name,
        first_time_boolean,
        folder->folder_attribute_list,
        relation_mto1_list(),
        security_entity,
        drillthru_dictionary );

int folder_attribute_calendar_date_name_list_length(
    folder->folder_attribute_list );

char *document_head_calendar_setup_string(
    folder_attribute_calendar_date_name_list_length() );

char *drillthru_document_title_string(
    char *state,
    char *base_folder_name );

char *drillthru_document_sub_title_string(
    char *drillthru_current_folder_name );

DOCUMENT *document =
    document_new(
        application_name,
        application_title_string( application_name ),
        drillthru_document_title_string(),
        drillthru_document_sub_title_string(),
        (char *)0 /* sub_sub_title_string */,
        folder->notepad,
        (char *)0 /* onload_javascript_string */,
        (menu) ? menu->html : (char *)0,
        document_head_menu_setup_string(
            (menu) ? 1 : 0 /* menu_boolean */ ),
        document_head_calendar_setup_string(),
        document_head_javascript_string()
            /* javascript_include_string */ );

char *document_form_html(
    document->html,
    document->document_head->html,
    document_head_close_tag(),
    document->document_body->html,
    form_drillthru->form_prompt_lookup_html,
    document_body_close_tag(),
    document_close_tag() );

free( form_drillthru->form_prompt_lookup_html );
```

DRILLTHRU_DOCUMENT (2)



```
/* Usage */
char *widget_table_edit_row_html(
    char *table_edit_state,
    int row_number,
    LIST *query_row_cell_list,
    RELATION_JOIN_ROW *relation_join_row,
    LIST *table_edit_apply_widget_container_list(),
    char *form_background_color(),
    boolean hidden_only );

/* Process */
char html[];
char *ptr = html;

*ptr = '\0';

for WIDGET_CONTAINER *widget_container in
    table_edit_apply_widget_container_list
{
    if (hidden_only
    && widget_container->widget_type != hidden )
    {
        continue;
    }
    else
    if ( !hidden_only
    && widget_container->widget_type == hidden )
    {
        continue;
    }

    char *widget_container_html(
        table_edit_state,
        row_number,
        query_row_cell_list,
        relation_join_row,
        form_background_color,
        widget_container );

    if ( !widget_container_html() ) continue;

    if ( ptr != html ) ptr += sprintf( ptr, "\n" );

    ptr += sprintf(
        ptr,
        "%s",
        widget_container_html() );

    free( widget_container_html() );
}

char *strdup( html );

/* Usage */
LIST *widget_hidden_container_list(
    char *dictionary_separate_prefix,
    DICTIONARY *dictionary );

/* Process */
LIST *widget_container_list = list_new();

LIST *dictionary_key_list( dictionary );

for char *key in dictionary_key_list()
{
    char *widget_hidden_key(
        char *dictionary_separate_prefix,
        char *key );

    WIDGET_CONTAINER *widget_container =
        widget_container_new(
            hidden,
            widget_hidden_key()
                /* widget_name */ );

    char *dictionary_get(
        key,
        dictionary );

    widget_container->hidden->value_string =
        dictionary_get();

    list_set(
        widget_container_list,
        widget_container );
}

/* Usage */
char *widget_dictionary_hidden_html(
    DICTIONARY *dictionary );

/* Process */
LIST *widget_container_dictionary_hidden_list(
    dictionary );

char *widget_container_list_hidden_html(
    -1 /* row_number */,
    widget_container_dictionary_hidden_list() );
```

WIDGET_CONTAINER (1)

```
/* Usage */
WIDGET_CONTAINER *widget_container_new(
    enum widget_type widget_type,
    char *widget_name );

/* Process */
WIDGET_CONTAINER *widget_container_calloc(
    void );

if ( widget_type == table_open )
    WIDGET_TABLE_OPEN *table_open =
        widget_table_open_calloc();
else
if ( widget_type == table_heading )
    WIDGET_TABLE_HEADING *table_heading =
        widget_table_heading_new(
            widget_name /* heading_string */ );
else
if ( widget_type == table_row )
    WIDGET_TABLE_ROW *table_row =
        widget_table_row_calloc();
else
if ( widget_type == table_close )
    WIDGET_TABLE_CLOSE *table_close =
        widget_table_close_calloc();
else
if ( widget_type == checkbox )
    WIDGET_CHECKBOX *checkbox =
        widget_checkbox_new(
            widget_name );
else
if ( widget_type == drop_down )
{
    WIDGET_DROP_DOWN *drop_down =
        widget_drop_down_new(
            widget_name );
}
else
if ( widget_type == multi_drop_down )
{
    WIDGET_MULTI_DROP_DOWN *multi_drop_down =
            widget_multi_drop_down_new(
                widget_name );
}
else
if ( widget_type == button )
    WIDGET_BUTTON *widget_button =
        widget_button_new(
            widget_name /* label */ );
else
if ( widget_type == character )
{
    WIDGET_TEXT *text =
        widget_text_new( widget_name );

    text->attribute_width_max_length =
        WIDGET_TEXT_DEFAULT_MAX_LENGTH;

    text->widget_text_display_size =
        WIDGET_TEXT_DEFAULT_DISPLAY_SIZE;
}
else
if ( widget_type == widget_date )
{
    WIDGET_DATE *date =
        widget_date_new( widget_name );
}
else
if ( widget_type == widget_time )
{
    WIDGET_TIME *time =
        widget_time_new( widget_name );
}
else
if ( widget_type == non_edit_text )
{
    WIDGET_NON_EDIT_TEXT *non_edit_text =
        widget_non_edit_text_new(
            widget_name );
}
else
if ( widget_type == yes_no )
{
    WIDGET_YES_NO *yes_no =
        widget_yes_no_new(
            widget_name );
}
else
if ( widget_type == hidden )
{
    WIDGET_HIDDEN *hidden =
        widget_hidden_calloc();

    hidden->widget_name = widget_name;
}
else
if ( widget_type == line_break )
{
    WIDGET_LINE_BREAK *line_break =
        widget_line_break_new();
}
else
if ( widget_type == table_data )
{
    WIDGET_TABLE_DATA *table_data =
        widget_table_data_calloc();
}
else
if ( widget_type == notepad )
{
    WIDGET_NOTEPAD *notepad =
        widget_notepad_new( widget_name );
}
else
if ( widget_type == encrypt )
{
    WIDGET_ENCRYPT *encrypt =
        widget_encrypt_new(
            widget_name );
}
else
/*
if ( widget_type == reference_number )
{
    WIDGET_REFERENCE_NUMBER *
        reference_number =
            widget_reference_number_calloc();

    reference_number->widget_name = widget_name;
}
*/
```

WIDGET_CONTAINER (2)

```
/* Process (continued) */
else
if ( widget_type == upload )
{
    WIDGET_UPLOAD *upload =
        widget_upload_new(
            widget_name );
}
else
if ( widget_type == blank )
{
    /* do nothing */
}
else
if ( widget_type == anchor )
{
    WIDGET_ANCHOR *anchor =
        widget_anchor_calloc();
}
else
if ( widget_type = unordered_list_open )
{
    /* do nothing */
}
else
if ( widget_type == unordered_list_close )
{
    /* do nothing */
}
else
if ( widget_type == list_item )
{
    /* do nothing */
}
else
if ( widget_type == widget_radio )
{
    WIDGET_RADIO *radio =
        widget_radio_new(
            widget_name );
}
else
if ( widget_type == widget_password )
{
    WIDGET_PASSWORD *password =
        widget_password_new(
            widget_name
                /* widget_password_display_name */ );
}

/* Attributes */
enum widget_type widget_type;
char *widget_name;

/* Set externally */
char *heading_string;
char *widget_key;
boolean  recall_boolean;

union {
WIDGET_TABLE_OPEN *table_open;
WIDGET_TABLE_HEADING *table_heading;
WIDGET_TABLE_ROW *table_row;
WIDGET_TABLE_CLOSE *table_close;
WIDGET_CHECKBOX *checkbox;
WIDGET_PROMPT_DROP_DOWN *prompt_drop_down;
WIDGET_DROP_DOWN *drop_down;
WIDGET_BUTTON *widget_button;
WIDGET_NON_EDIT_TEXT *non_edit_text;
WIDGET_HIDDEN *hidden;
WIDGET_LINE_BREAK *line_break;
WIDGET_TABLE_DATA *table_data;
WIDGET_MULTI_DROP_DOWN *multi_drop_down;
WIDGET_NOTEPAD *notepad;
WIDGET_TEXT *text;
WIDGET_ENCRYPT *encrypt;
WIDGET_YES_NO *yes_no;
WIDGET_DATE *date;
WIDGET_TIME *time;
WIDGET_UPLOAD *upload;
WIDGET_ANCHOR *anchor;
WIDGET_RADIO *radio;
WIDGET_AJAX *ajax;
WIDGET_PASSWORD *password;
};
```

enum widget_type widget_type

table_open	multi_drop_down	notepad	upload	widget_radio
table_heading	button	character	blank	widget_ajax
table_row	non_edit_text	encrypt	anchor	widget_password
table_close	hidden	yes_no	unordered_list_open	
checkbox	line_break	widget_date	unordered_list_close	
drop_down	table_data	widget_time	list_item	

WIDGET_CONTAINER (3)

```
/* Usage */
static char *widget_container_key(
    char *widget_name,
    int row_number );

/* Process */
if ( row_number > 0 )
{
    sprintf( static key[],
        "%s_%d",
        widget_name,
        row_number );
}
else
{
    strcpy( static key[], widget_name );
}

/* Usage */
static char *widget_container_name(
    char *attribute_name,
    int row_number );

/* Process */
static char *widget_container_key(
    attribute_name /* widget_name */,
    row_number );

/* Usage */
char *widget_container_value(
    char ATTRIBUTE_MULTI_KEY_DELIMITER,
    LIST *query_row_cell_list,
    char *widget_name );

/* Process */
if ( ATTRIBUTE_MULTI_KEY_DELIMITER )
{
    char *widget_container_multi_key_value(
        ATTRIBUTE_MULTI_KEY_DELIMITER,
        query_row_cell_list,
        widget_name );
}
else
{
    QUERY_CELL *query_cell_seek(
        widget_name,
        query_row_cell_list );

    if ( !query_cell_seek() ) return NULL;

    query_cell_seek()->display_datum );
}

/* Usage */
char *widget_container_multi_key_value(
    char ATTRIBUTE_MULTI_KEY_DELIMITER,
    LIST *query_row_cell_list,
    char *widget_name );

/* Process */
char *ptr = container_value[];

LIST *list_string_to_list(
    widget_name /* list_string */,
    ATTRIBUTE_MULTI_KEY_DELIMITER );

for char *widget_name in list_string_to_list()
{
    QUERY_CELL *query_cell_seek(
        widget_name,
        query_row_cell_list );

    if ( !query_cell_seek() ) return NULL;

    if ( ptr != container_value )
    {
        ptr += sprintf( ptr,
            "%c",
            ATTRIBUTE_MULTI_KEY_DELIMITER );
    }

    ptr += sprintf( ptr,
        "%s",
        query_cell_seek()->display_datum );
}

void list_free_string_list( list_string_to_list() );

/* Usage */
char *widget_container_list_html(
    char *state,
    int row_number /* -1 will omit index suffix */,
    char *background_color,
    LIST *widget_container_list );

/* Process */
char html[];
char *ptr = html;

for WIDGET_CONTAINER *widget_container in
    widget_container_list
{
    if ( widget_container->widget_type == hidden )
        continue;

    char *widget_container_html(
        state,
        row_number,
        (LIST *)0 /* query_row_cell_list */,
        (RELATION_JOIN_ROW *)0,
        background_color,
        widget_container );

    if ( ptr != html ) ptr += sprintf( ptr, "\n" );

    ptr += sprintf( ptr,
        "%s",
        widget_container_html() );

    free( widget_container_html() );
}
```

WIDGET_CONTAINER (4)

```
/* Usage */
char *widget_container_html(
    char *state,
    int row_number /* -1 will omit index suffix */,
    LIST *query_row_cell_list,
    RELATION_JOIN_ROW *relation_join_row,
    char *background_color,
    WIDGET_CONTAINER *widget_container );

/* Process */
if ( widget_container->widget_type == table_open )
{
    return
    char *strdup(
        widget_table_open_tag(
            widget_container->
                table_open->
                border_boolean,
            widget_container->
                table_open->
                cell_spacing,
            widget_container->
                table_open->
                cell_padding ) );
}
else
if ( widget_container->widget_type == table_heading )
{
    return
    char *widget_table_heading_tag(
        widget_container->
            table_heading->
            heading_string );
}
else
if ( widget_container->widget_type == table_row )
{
    return
    char *strdup(
        widget_table_row_tag(
            background_color ) );
}
else
if ( widget_container->widget_type == table_close )
{
    return
    char *strdup(
        widget_table_close_tag() );
}
else
if ( widget_container->widget_type == checkbox )
{
    return
    char *widget_container_checkbox_html(
        char *state,
        int row_number,
        LIST *query_row_cell_list,
        char *background_color,
        WIDGET_CHECKBOX *
            widget_container->checkbox );
}
else
if ( widget_container->widget_type == drop_down )
{
    return
    char *widget_container_drop_down_html(
        state,
        row_number,
        query_row_cell_list,
        relation_join_row,
        background_color,
        widget_container->drop_down );
}
else
if (widget_container->widget_type ==
    multi_drop_down )
{
    return
    char *widget_container_multi_drop_down_html(
        char *state,
        WIDGET_MULTI_DROP_DOWN *
            widget_container->multi_drop_down );
}
else
if ( widget_container->widget_type == character )
{
    return
    char *widget_container_text_html(
        state,
        row_number,
        query_row_cell_list,
        row_number,
        background_color,
        widget_container->text );
}
else
if ( widget_container->widget_type == hidden )
{
    return
    char *widget_hidden_html(
        row_number,
        query_row_cell_list,
        widget_container->hidden );
}
else
if ( widget_container->widget_type == notepad )
{
    return
    char *widget_container_notepad_html(
        char *state,
        int row_number,
        LIST *query_row_cell_list,
        char *background_color,
        WIDGET_NOTEPAD *widget_container->notepad );
}
else
if ( widget_container->widget_type == upload )
{
    return
    char *widget_container_upload_html(
        query_row_cell_list,
        background_color,
        widget_container->upload );
}
else
if ( widget_container->widget_type == yes_no )
{
    return
    char *widget_container_yes_no_html(
        char *state,
        int row_number,
        LIST *query_row_cell_list,
        char *background_color,
        WIDGET_YES_NO *widget_container->yes_no );
}
else
if ( widget_container->widget_type == widget_date )
{
    return
    char *widget_container_date_html(
        state,
        row_number,
        query_row_cell_list,
        background_color,
        widget_container->date );
}
else
```

WIDGET_CONTAINER (5)

```
/* widget_container_html() (continued) */
if ( widget_container->widget_type == widget_time )
{
    return
    char *widget_container_time_html(
        char *state,
        int row_number,
        LIST *query_row_cell_list,
        char *background_color,
        WIDGET_TIME *widget_container->time );
}
else
if ( widget_container->widget_type == button )
{
    return
    char *widget_button_html(
        row_number,
        widget_container->widget_button );
}
else
if ( widget_container->widget_type == non_edit_text )
{
    return
    char *widget_container_non_edit_text_html(
        query_row_cell_list,
        widget_container->non_edit_text );
}
else
if ( widget_container->widget_type == line_break )
{
    return
    char *strdup(
        widget_line_break_html() );
}
else
if ( widget_container->widget_type == table_data )
{
    return
    char *strdup(
        widget_table_data_html(
            widget_container->table_data->align_right,
            widget_container->
                table_data->
                    column_span ) );
}
else
if ( widget_container->widget_type == encrypt )
{
    return
    char *widget_container_encrypt_html(
        int row_number,
        LIST *query_row_cell_list,
        WIDGET_ENCRYPT *
            widget_container->encrypt );
}
else
if ( widget_container->widget_type == blank )
{
    return strdup( "" );
}
else
if ( widget_container->widget_type == anchor )
{
    return
    char *widget_container_anchor_html(
        WIDGET_ANCHOR *widget_container->anchor );
}
else
if ( widget_container->widget_type ==
        unordered_list_open )
{
    return strdup( "<ul>" );
}
else
if ( widget_container->widget_type ==
        unordered_list_close )
{
    return strdup( "</ul>" );
}
else
if ( widget_container->widget_type == list_item )
{
    return strdup( "<li>" );
}
else
if ( widget_container->widget_type == widget_radio )
{
    return
    char *widget_container_radio_html(
        WIDGET_RADIO *widget_container->radio );
}
else
if ( widget_container->widget_type == widget_ajax )
{
    return
    char *widget_container_ajax_html(
        widget_container->
            ajax->
            ajax_client_widget_container_list,
        query_row_cell_list,
        row_number,
        background_color );
}
else
if (    widget_container->widget_type ==
        widget_password )
{
    return
    char *widget_password_html(
        widget_container->
            password->
            attribute_name,
        widget_container->
            password->
            display_name,
        widget_container->
            password->
            text
            /* widget_text */,
        state,
        row_number,
        query_row_cell_list,
        background_color );
}
else
{
    exit( 1 );
}
```

WIDGET_CONTAINER (6)

```
/* Usage */
char *widget_container_list_hidden_html(
    int row_number,
    LIST *widget_container_list );

/* Process */
char html[];
char *ptr = html;

for WIDGET_CONTAINER *widget_container in
    widget_container_list
{
    if ( widget_container->widget_type != hidden )
        continue;

    char *widget_container_html(
        (char *)0 /* state */,
        row_number,
        (LIST *)0 /* query_row_cell_list */,
        (RELATION_JOIN_ROW *)0,
        (char *)0 /* background_color */,
        widget_container );

    if ( !widget_container_html() ) continue;

    if ( ptr != html ) ptr += sprintf( ptr, "\n" );

    ptr += sprintf( ptr,
        "%s",
        widget_container_html() );

    free( widget_container_html() );
}

/* Usage */
char *widget_container_non_edit_text_html(
    LIST *query_row_cell_list,
    WIDGET_NON_EDIT_TEXT *non_edit_text );

/* Process */
char *widget_non_edit_text_html(
    query_row_cell_list,
    widget_name,
    non_edit_text->no_initial_capital_boolean );

/* Usage */
char *widget_container_text_html(
    char *state,
    int row_number,
    LIST *query_row_cell_list,
    int row_number,
    char *background_color,
    WIDGET_TEXT *widget_text );

/* Process */
char *widget_text_html(
    widget_text->widget_name,
    widget_text->attribute_width_max_length,
    widget_text->widget_text_display_size,
    widget_text->null_to_slash,
    widget_text->prevent_carrot_boolean,
    widget_text->on_change,
    widget_text->tab_order,
    widget_text->value_string,
    widget_text->viewonly,
    state,
    row_number,
    query_row_cell_list,
    background_color );

/* Usage */
char *widget_container_date_html(
    char *state,
    int row_number,
    LIST *query_row_cell_list,
    char *background_color,
    WIDGET_DATE *widget_container->date );

/* Process */
char *widget_date_html(
    date->widget_name,
    date->datatype_name,
    date->attribute_width_max_length,
    date->display_size,
    date->null_to_slash_boolean,
    date->validate_date_boolean,
    date->post_change_javascript,
    date->tab_order,
    date->application_name,
    date->login_name,
    state,
    row_number,
    query_row_cell_list,
    background_color );

/* Usage */
char *widget_container_ajax_html(
    LIST *widget_container->
        ajax->
        ajax_client_widget_container_list,
    LIST *query_row_cell_list,
    int row_number,
    char *background_color );

/* Process */
char *widget_ajax_html(
    ajax_client_widget_container_list,
    query_row_cell_list,
    row_number,
    background_color );
```

WIDGET_CONTAINER (7)

```
/* Usage */
LIST *widget_container_dictionary_hidden_list(
    DICTIONARY *dictionary );

/* Process */
LIST *key_list = dictionary_key_list( dictionary );

for char *key in key_list
{
    char *data = dictionary_get( key, dictionary );

    list_set(
        widget_container_list,
        ( WIDGET_CONTAINER *widget_container =
            widget_container_new(
                hidden,
                key ) ) );

    widget_container->hidden->value_string = data;
}

/* Usage */
WIDGET_CONTAINER *
    widget_lookup_checkbox_container(
        char *checkbox_label );

/* Process */
WIDGET_CONTAINER *widget_container =
    widget_container_new(
        checkbox,
        checkbox_label );

widget_container->checkbox->prompt = "Lookup";

/* Usage */
WIDGET_CONTAINER *
    widget_confirm_checkbox_container(
        char *checkbox_label,
        char *checkbox_prompt );

/* Process */
WIDGET_CONTAINER *widget_container =
    widget_container_new(
        checkbox /* widget_type */,
        checkbox_label /* widget_name */ );

widget_container->checkbox->prompt =
    checkbox_prompt;

/* Usage */
WIDGET_CONTAINER *
    widget_submit_button_container(
        char *form_name );

/* Process */
WIDGET_CONTAINER *submit_widget_container =
    widget_container_new(
        button, (char *)0 );

submit_widget_container->widget_button->button =
    button_submit(
        form_name,
        (char *)0 /* multi_select_all_javascript */,
        (char *)0 /* recall_save_javascript */,
        (char *)0 /* verify_widths_javascript */ );

/* Usage */
char *widget_container_drop_down_html(
    char *state,
    int row_number,
    LIST *query_row_cell_list,
    RELATION_JOIN_ROW *relation_join_row,
    char *background_color,
    WIDGET_DROP_DOWN *
        widget_container->drop_down );

/* Process */
char *widget_drop_down_id(
    drop_down->many_folder_name,
    drop_down->widget_name );

char *widget_drop_down_html(
    drop_down->widget_name
        /* mutually exclusive */,
    drop_down->attribute_name_list
        /* mutually exclusive */,
    drop_down->top_select_boolean,
    drop_down->null_boolean,
    drop_down->not_null_boolean,
    drop_down->bottom_select_boolean,
    drop_down->display_size,
    drop_down->tab_order,
    drop_down->multi_select_boolean,
    drop_down->post_change_javascript,
    state,
    row_number,
    query_row_cell_list,
    relation_Join_row,
    drop_down->widget_drop_down_option_list,
    drop_down->foreign_key_list,
    background_color,
    widget_drop_down_id() );

/* Usage */
char *widget_container_heading_html(
    LIST *widget_container_list );

/* Process */

/* Usage */
LIST *widget_container_back_to_top_list(
    void );

/* Process */
LIST *widget_container_list = list_new();

list_set(
    widget_container_list,
    widget_container_new(
        table_open, (char *)0 ) );

list_set(
    widget_container_list,
    widget_container_new(
        table_row, (char *)0 ) );

list_set(
    widget_container_list,
    widget_container_new(
        table_data, (char *)0 ) );

list_set(
    widget_container_list,
    ( WIDGET_CONTAINER *widget_container =
        widget_container_new(
            button, (char *)0 ) ) );

widget_container->
    widget_button->
    button = button_to_top();

list_set(
    widget_container_list,
    widget_container_new(
        table_close, (char *)0 ) );
```

WIDGET_CONTAINER (8)

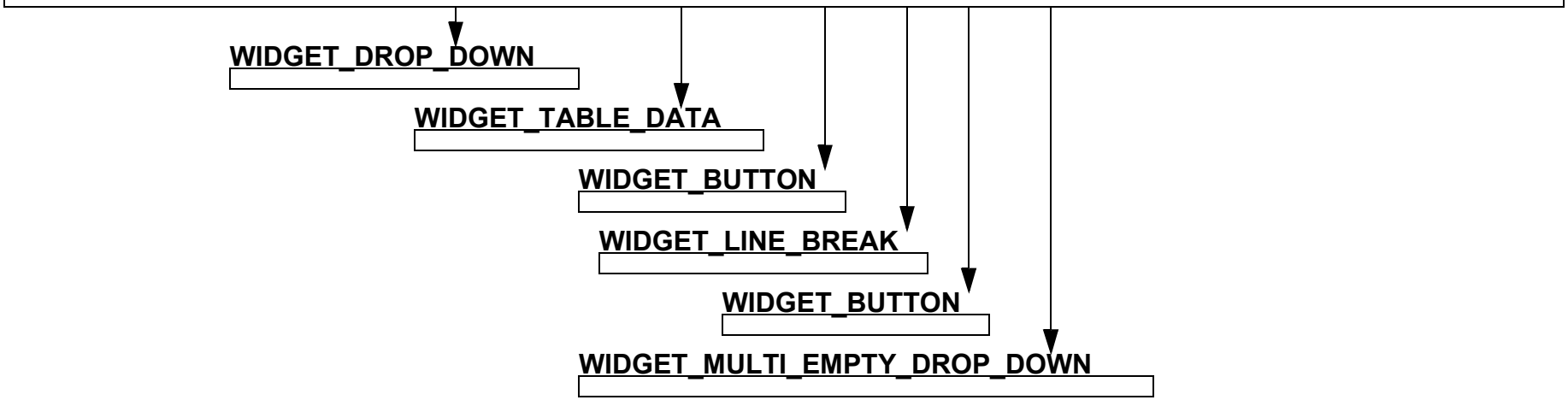
```
/* Usage */
char *widget_container_upload_html(
    LIST *query_row_cell_list,
    char *background_color,
    WIDGET_UPLOAD *widget_container->upload );

/* Process */
if ( !list_length( query_row_cell_list ) )
{
    char *widget_upload_prompt_frame_htm_stringl(
        widget_upload->widget_name,
        widget_upload->recall_widget_name,
        widget_upload->tab_order );
}
else
{
    char *widget_upload_edit_frame_html(
        widget_upload->application_name,
        widget_upload->session_key,
        widget_upload->widget_name,
        query_row_cell_list,
        background_color );
}

/* Usage */
LIST *widget_container_heading_label_list(
    LIST *widget_container_list );

/* Process */
LIST *heading_label_list = list_new();

for WIDGET_CONTAINER *widget_container in
    widget_container_list
{
    if ( widget_container->heading_string )
    {
        list_set(
            heading_label_list,
            widget_container->heading_string );
    }
}

/* Usage */
LIST *widget_container_heading_list(
    LIST *widget_container_heading_label_list() );

/* Process */
LIST *widget_container_list = list_new();

list_set(
    widget_container_list,
    widget_container_new( table_row, (char *)0 );

for char *heading_label in
    widget_container_heading_label_list
{
    list_set(
        widget_container_list,
        widget_container_new(
            table_heading,
            strdup(
                string_initial_capital(
                    heading_string[],
                    heading_label ) ) ) ) );
}

/* Usage */
LIST *widget_container_heading_name_list(
    LIST *widget_container_list );

/* Process */
for WIDGET_CONTAINER *widget_container in
    widget_container_list
{
    if ( widget_container->heading_string )
    {
        list_set(
            heading_name_list,
            widget_container->widget_name );
    }
}

/* Usage */
WIDGET_CONTAINER *widget_container_ajax_new(
    char *widget_name,
    LIST *ajax_client_widget_container_list );

/* Process */
WIDGET_CONTAINER *widget_container =
    widget_container_calloc();

widget_container->widget_name = widget_name;
widget_container->widget_type = widget_ajax;
widget_container->ajax =
    widget_ajax_new(
        ajax_client_widget_container_list );

/* Usage */
boolean widget_container_recall_boolean(
    boolean widget_container->recall_boolean );

/* Process */

/* Usage */
void widget_containter_list_validate_date_unset(
    LIST *widget_container_list );

/* Process */

/* Usage */
int widget_container_tab_order( void );

/* Process */

/* Usage */
static char *widget_container_tab_order_html(
    int widget_container_tab_order() );

/* Process

/* Usage */
static char *widget_container_javascript_html(
    char *javascript_replace() );

/* Process */

/* Usage */
char *widget_container_background_color_html(
    char *background_color );

/* Process */

/* Usage */
char *widget_container_number_commas_string(
    char *widget_name,
    char *value );

/* Process */
```

WIDGET_MULTI_DROP_DOWN (1)

```
/* Usage */
WIDGET_MULTI_DROP_DOWN *
   widget_multi_drop_down_new(
      char *widget_name );

/* Process */
WIDGET_MULTI_DROP_DOWN *
   widget_multi_drop_down_calloc(
      void );

char *widget_multi_drop_down_original_name(
   char *WIDGET_MULTI_ORIGINAL_PREFIX,
   char *widget_name );

WIDGET_DROP_DOWN *original_drop_down =
   widget_drop_down_calloc()

original_drop_down->
   widget_name =
      widget_multi_drop_down_original_name();

original_drop_down->
   display_size = WIDGET_MULTI_DISPLAY_SIZE;

WIDGET_TABLE_DATA *table_data =
   widget_table_data_calloc();

char *widget_multi_drop_down_right_label(
   void );

WIDGET_BUTTON *move_right_button =
   widget_button_new(
      widget_multi_drop_down_right_label() );

char *widget_multi_drop_down_right_action_string(
   char *widget_multi_drop_down_original_name(),
   char *widget_name,
   char WIDGET_MULTI_LEFT_RIGHT_DELIMITER );

move_right_button->button->action_string =
   widget_multi_drop_down_right_action_string();

move_right_button->button->hover_message =
   WIDGET_MULTI_DROP_DOWN_RIGHT_MESSAGE;

move_right_button->button->html =
   widget_button_html(
      0 /* row_number */,
      move_right_button /* WIDGET_BUTTON * */ );

WIDGET_LINE_BREAK *widget_line_break
    widget_line_break_new();

char *widget_multi_drop_down_left_label( void );

WIDGET_BUTTON *move_left_button =
   widget_button_new(
      widget_multi_drop_down_left_label() );

char *widget_multi_drop_down_left_action_string(
   char *widget_multi_drop_down_original_name(),
   char *widget_name,
   char WIDGET_MULTI_LEFT_RIGHT_DELIMITER );

move_left_button->button->action_string =
      widget_multi_drop_down_left_action_string();

move_left_button->button->hover_message =
      WIDGET_MULTI_DROP_DOWN_LEFT_MESSAGE;

move_left_button->button->html =
   widget_button_html(
      0 /* row_number */,
      move_left_button /* WIDGET_BUTTON * */ );

WIDGET_MULTI_EMPTY_DROP_DOWN *
   multi_empty_drop_down =
      widget_multi_empty_drop_down_new(
         widget_name );
```

WIDGET_MULTI_DROP_DOWN (2)

```
/* Usage */
char *widget_multi_drop_down_html(
    char *state,
    WIDGET_DROP_DOWN *original_drop_down,
    WIDGET_TABLE_DATA *table_data,
    WIDGET_BUTTON *move_right_button,
    WIDGET_LINE_BREAK *widget_line_break,
    WIDGET_BUTTON *move_left_button,
    WIDGET_MULTI_EMPTY_DROP_DOWN *
        multi_empty_drop_down,
    boolean null_boolean,
    boolean not_null_boolean );

/* Process */
char *widget_drop_down_html(
    original_drop_down->widget_name,
    (LIST *)0 /* attribute_name_list */,
    0 /* not top_select_boolean */,
    null_boolean,
    not_null_boolean,
    0 /* not bottom_select_boolean */,
    WIDGET_MULTI_DISPLAY_SIZE,
    0 /* tab_order */,
    1 /* multi_select_boolean */,
    original_drop_down->post_change_javascript,
    state,
    -1 /* row_number to omit index suffix */,
    (LIST *)0 /* query_row_cell_list */,
    (RELATION_JOIN_ROW *)0,
    original_drop_down->widget_drop_down_option_list,
    (LIST *)0 /* foreign_key_list */,
    (char *)0 /* background_color */,
    (char *)0 /* widget_drop_down_id */ );

char *widget_table_data_tag(
    table_data->align_right,
    table_data->column_span,
    table_data->width_em,
    table_data->style );

char *widget_table_open_tag(
    0 /* not border_boolean */,
    0 /* cell_spacing */,
    0 /* cell_padding */ );

char *widget_table_row_tag(
    (char *)0 /* background_color */ );

char *widget_multi_drop_down_html_string(
    char *widget_drop_down_html(),
    char *widget_table_data_tag(),
    char *widget_table_open_tag(),
    char *widget_table_row_tag(),
    char *move_right_button->button->html,
    char *widget_line_break->tag,
    char *move_left_button->button->html,
    char *widget_table_close_tag(),
    char *widget_multi_empty_drop_down->html );

/* Attributes */
char *original_name;
WIDGET_DROP_DOWN *original_drop_down;
WIDGET_TABLE_DATA *table_data;
WIDGET_BUTTON *move_right_button;
WIDGET_LINE_BREAK *widget_line_break;
WIDGET_BUTTON *move_left_button;
WIDGET_MULTI_EMPTY_DROP_DOWN *
    multi_empty_drop_down;
boolean null_boolean;
boolean not_null_boolean;
```



WIDGET_DROP_DOWN (1)

```
/* Usage */
WIDGET_DROP_DOWN *widget_drop_down_new(
    char *widget_name );

/* Process */
WIDGET_DROP_DOWN *widget_drop_down_calloc(
    void );

/* Usage */
char *widget_drop_down_html(
    char *widget_name /* mutually exclusive */,
    LIST *attribute_name_list /* mutually exclusive */,
    boolean top_select_boolean,
    boolean null_boolean,
    boolean not_null_boolean,
    boolean bottom_select_boolean,
    int display_size,
    int tab_order,
    boolean multi_select_boolean,
    char *post_change_javascript,
    char *state,
    int row_number,
    LIST *query_row_cell_list,
    RELATION_JOIN_ROW *relation_join_row,
    LIST *widget_drop_down_option_list,
    LIST *foreign_key_list,
    char *background_color,
    char *widget_drop_down_id() );

/* Process */
char *widget_drop_down_name(
    const char ATTRIBUTE_MULTI_KEY_DELIMITER,
    char *widget_name /* mutually exclusive */,
    LIST *attribute_name_list /* mutually exclusive */ );

int widget_drop_down_display_size(
    int WIDGET_MULTI_DISPLAY_SIZE,
    int display_size,
    boolean multi_select_boolean );

char *widget_container_key(
    widget_drop_down_name(),
    row_number );

char *widget_drop_down_row_number_id(
    char *widget_drop_down_id,
    int row_number,
    widget_container_key() );

if ( list_length( query_row_cell_list ) )
{
    char *widget_drop_down_value(
        ATTRIBUTE_MULTI_KEY_DELIMITER,
        widget_name,
        query_row_cell_list,
        relation_join_row,
        foreign_key_list );
}

if ( post_change_javascript )
{
    char *javascript_replace(
        post_change_javascript,
        state,
        row_number ),
}

boolean widget_drop_down_top_select_boolean(
    boolean top_select_boolean,
    char *widget_drop_down_value() );

char *widget_drop_down_select_tag(
    widget_drop_down_display_size(),
    tab_order,
    multi_select_boolean,
    widget_container_key(),
    widget_drop_down_row_number_id(),
    javascript_replace(),
    background_color,
    widget_drop_down_top_select_boolean() );

if ( relation_join_row )
{
    char *widget_drop_down_relation_join_html(
        widget_name,
        relation_join_row );
}

if ( !widget_drop_down_relation_join_html() )
{
    char *widget_drop_down_option_list_html(
        widget_drop_down_option_list,
        widget_drop_down_value() );
}

boolean widget_drop_down_bottom_select_boolean(
    boolean bottom_select_boolean,
    boolean widget_drop_down_top_select_boolean() );

char *widget_drop_down_select_close_tag(
    null_boolean /* widget_drop_down_null_boolean() */,
    not_null_boolean,
    widget_drop_down_bottom_select_boolean() );

char *widget_drop_down_html_string(
    char *widget_drop_down_select_tag(),
    char *widget_drop_down_relation_join_html(),
    char *widget_drop_down_option_list_html(),
    char *widget_drop_down_select_close_tag() );

free( widget_drop_down_select_tag );

if ( widget_drop_down_relation_join_html() )
        free( widget_drop_down_relation_join_html() );

if ( widget_drop_down_option_list_html() )
        free( widget_drop_down_option_list_html() );

return
widget_drop_down_html_string();
```

WIDGET_DROP_DOWN (2)

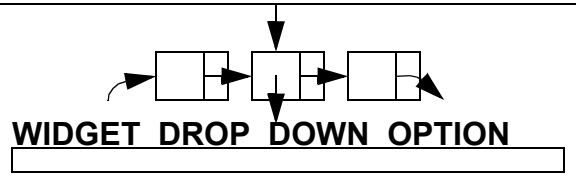
```
/* Usage */
char *widget_drop_down_value(
    const char ATTRIBUTE_MULTI_KEY_DELIMITER,
    char *widget_name,
    LIST *query_row_cell_list,
    RELATION_JOIN_ROW *relation_join_row,
    LIST *foreign_key_list );

/* Process */
static char drop_down_value[];
char *ptr = drop_down_value;

LIST *relation_join_row_folder_name_list(
    relation_join_row );

boolean list_string_exists(
    widget_name,
    relation_join_row_folder_name_list() );

if ( list_string_exists() ) return NULL;

*drop_down_value = '\0';

for char *foreign_key in foreign_key_list
{
    QUERY_CELL *query_cell_seek(
        foreign_key /* attribute_name */,
        query_cell_list );

    if ( ptr != drop_down_value )
    {
        ptr += sprintf(
                    ptr,
                    "%c",
                    ATTRIBUTE_MULTI_KEY_DELIMITER );
    }

    ptr += sprintf( ptr,
        "%s",
        query_cell->select_datum );
}

if ( !*drop_down_value )
    return NULL;
else
    return drop_down_value;

/* Usage */
char *widget_drop_down_select_tag(
    int widget_drop_down_display_size(),
    int tab_order,
    boolean multi_select_boolean,
    char *widget_container_key(),
    char *widget_drop_down_row_number_id(),
    char *javascript_replace(),
    char *background_color,
    boolean top_select_boolean );

/* Process */
char tag[];
char *ptr = tag;

ptr += sprintf( ptr,
    "<select name=\"%s\"",
    widget_container_key );

if ( widget_drop_down_row_number_id )
{
    ptr += sprintf( ptr,
        " id=\"%s\"",
        widget_drop_down_row_number_id );
}

ptr += sprintf( ptr, " size=%d", display_size );

if ( multi_select_boolean )
{
    ptr += sprintf( ptr, " multiple" );
}

if ( tab_order > 0 )
{
    ptr += sprintf( ptr,
        "%s",
        static char *widget_container_tab_order_html(
            tab_order ) );
}

if ( background_color )
{
    ptr += sprintf( ptr,
        "%s",
        widget_container_background_color_html(
            background_color ) );
}

if ( javascript_replace )
{
    ptr += sprintf( ptr,
        "%s",
        widget_container_javascript_html(
            javascript_replace ) );
}

ptr += sprintf( ptr, ">" );

if ( top_select_boolean )
{
    sprintf( ptr,
        "\n%s",
        widget_drop_down_option_static_tag(
            WIDGET_SELECT_OPERATOR ) );
}
```

WIDGET_DROP_DOWN (3)

```
/* Usage */
char *widget_drop_down_select_close_tag(
    boolean null_boolean,
    boolean not_null_boolean,
    boolean bottom_select_boolean );

/* Process */
char static tag[];
char *ptr = tag;

if ( null_boolean )
{
    ptr += sprintf( ptr,
        "%s\n",
        widget_drop_down_option_static_tag(
            QUERY_IS_NULL ) );
}

if ( not_null_boolean )
{
    ptr += sprintf( ptr,
        "%s\n",
        widget_drop_down_option_static_tag(
            QUERY_NOT_NULL ) );
}

if ( bottom_select_boolean )
{
    ptr += sprintf( ptr,
        "%s\n",
        widget_drop_down_option_static_tag(
            WIDGET_SELECT_OPERATOR ) );
}

sprintf( ptr, "</select>" );
```

```
/* Usage */
char *widget_drop_down_relation_join_html(
    char *widget_name,
    RELATION_JOIN_ROW *relation_join_row );

/* Process */
char html[];
char *ptr = html

if ( !relation_join_row ) return NULL;

RELATION_JOIN_FOLDER *relation_join_folder_seek(
    widget_name /* folder_name */,
    relation_join_row->relation_join_folder_list );

if ( !relation_join_folder_seek() ) return NULL;

for char *delimited_string in
    relation_join_folder_seek()->delimited_list
{
    char *string_initial_capital(
        buffer[],
        delimited_string );

    char *widget_drop_down_option_tag(
        delimited_string /* value_string */,
        string_initial_capital() /* display_string */ );

    if ( ptr != html ) ptr += sprintf( ptr, "\n" );

    ptr += sprintf( ptr,
            "%s",
            widget_drop_down_option_tag() );
}
```

```
/* Usage */
char *widget_drop_down_id(
    char *widget_drop_down->many_folder_name,
    char *widget_drop_down->widget_name );

/* Process */
if ( many_folder_name && widget_name )
{
    snprintf(
        static char widget_id[],
        sizeof ( widget_id ),
        "%s_%s",
        many_folder_name,
        widget_name );
}

/* Attributes */
char *widget_name /* mutually exclusive */;

/* Set externally */
LIST *attribute_name_list /* mutually exclusive */;
char *many_folder_name;
boolean  top_select_boolean;
boolean null_boolean;
boolean not_null_boolean;
boolean bottom_select_boolean;
int display_size;
int tab_order;
boolean multi_select_boolean;
char *post_change_javascript;
LIST *widget_drop_down_option_list;
LIST *foreign_key_list;
```



WIDGET_MULTI_EMPTY_DROP_DOWN

```
/* Usage */
WIDGET_MULTI_EMPTY_DROP_DOWN *
    widget_multi_empty_drop_down_new(
        char *widget_name );

/* Process */
WIDGET_MULTI_EMPTY_DROP_DOWN *
    widget_multi_empty_drop_down_calloc(
        void );

char *widget_drop_down_select_tag(
    WIDGET_MULTI_DISPLAY_SIZE,
    0 /* tab_order */,
    1 /* multi_select */,
    widget_name /* widget_container_key */,
    (char *)0 /* widget_drop_down_row_number_id */,
    (char *)0 /* javascript_replace */,
    (char *)0 /* background_color */,
    0 /* not top_select_boolean */ );

char *widget_drop_down_select_close_tag(
    0 /* not null_boolean */,
    0 /* not not_null_boolean */,
    0 /* not bottom_select_boolean */ );

char *widget_multi_empty_drop_down_html(
    char *widget_drop_down_select_tag(),
    char *widget_drop_down_select_close_tag() );

/* Attributes */
char *widget_drop_down_select_tag;
char *widget_drop_down_select_close_tag;
char *html;
```

WIDGET_DROP_DOWN_OPTION (1)

```
/* Usage */
LIST *widget_drop_down_option_list(
    const char ATTRIBUTE_MULTI_KEY_DELIMITER,
    char WIDGET_DROP_DOWN_LABEL_DELIMITER
        /* probably '|' */,
    char WIDGET_DROP_DOWN_EXTRA_DELIMITER
        /* probably '[' */,
    char *WIDGET_DROP_DOWN_DASH_DELIMITER,
    LIST *delimited_list,
    boolean no_initial_capital );

/* Process */
LIST *list = list_new();

char *sql_delimiter_string(
    ATTRIBUTE_MULTI_KEY_DELIMITER );

for char *delimited_string in delimited_list
{
    list_set(
        list,
        WIDGET_DROP_DOWN_OPTION *
            widget_drop_down_option_delimited_new(
                WIDGET_DROP_DOWN_LABEL_DELIMITER,
                WIDGET_DROP_DOWN_EXTRA_DELIMITER,
                WIDGET_DROP_DOWN_DASH_DELIMITER,
                no_initial_capital,
                sql_delimiter_string(),
                delimited_string ) );
}

/* Usage */
WIDGET_DROP_DOWN_OPTION *
    widget_drop_down_option_delimited_new(
        char WIDGET_DROP_DOWN_LABEL_DELIMITER,
        char WIDGET_DROP_DOWN_EXTRA_DELIMITER,
        char *WIDGET_DROP_DOWN_DASH_DELIMITER,
        boolean no_initial_capital,
        char *sql_delimiter_string(),
        char *delimited_string );

/* Process */
WIDGET_DROP_DOWN_OPTION *
    widget_drop_down_option_calloc(
        void );

char *strdup(
    widget_drop_down_option_value_string(
        WIDGET_DROP_DOWN_LABEL_DELIMITER /* '|' */,
        WIDGET_DROP_DOWN_EXTRA_DELIMITER /* '[' */,
        delimited_string ) );

char *widget_drop_down_option_display_string(
    WIDGET_DROP_DOWN_LABEL_DELIMITER /* '|' */,
    WIDGET_DROP_DOWN_DASH_DELIMITER,
    no_initial_capital,
    sql_delimiter_string,
    delimited_string );

char *strdup(
    widget_drop_down_option_tag(
        widget_drop_down_option_value_string(),
        widget_drop_down_option_display_string() ) );

/* Usage */
char *widget_drop_down_option_value_string(
    char WIDGET_DROP_DOWN_LABEL_DELIMITER
        /* probably '|' */,
    char WIDGET_DROP_DOWN_EXTRA_DELIMITER
        /* probably '[' */,
    char *delimited_string );

/* Process */
if ( string_character_exists(
        delimited_string,
        WIDGET_DROP_DOWN_LABEL_DELIMITER ) )
{
    piece(
        static char value_string[],
        WIDGET_DROP_DOWN_LABEL_DELIMITER,
        delimited_string,
        0 );
}
else
if ( string_character_exists(
        delimited_string,
        WIDGET_DROP_DOWN_EXTRA_DELIMITER ) )
{
    piece(
        static char value_string[],
        WIDGET_DROP_DOWN_EXTRA_DELIMITER,
        delimited_string,
        0 );
        string_trim_right_spaces(
            value_string /* buffer */ );
}
else
{
    strcpy( static char value_string[], delimited_string );
}

/* Usage */
char *widget_drop_down_option_display_string(
    const char WIDGET_DROP_DOWN_LABEL_DELIMITER
        /* probably '|' */,
    const *WIDGET_DROP_DOWN_DASH_DELIMITER,
    boolean no_initial_capital,
    char *sql_delimiter_string,
    char *delimited_string );

/* Process */
if ( string_character_exists(
        delimited_string,
        WIDGET_DROP_DOWN_LABEL_DELIMITER ) )
{
    piece(
        char display_string[],
        WIDGET_DROP_DOWN_LABEL_DELIMITER,
        delimited_string,
        1 );
}
else
{
    strcpy( char display_string[], delimited_string );
}

string_search_replace(
    display_string
        /* source_destination */,
    sql_delimiter_string
        /* search_string */,
    WIDGET_DROP_DOWN_DASH_DELIMITER
        /* replace_string */ );

if ( !no_initial_capital )
{
    string_initial_capital(
        display_string,
        display_string );
}
```

WIDGET_DROP_DOWN_OPTION (2)

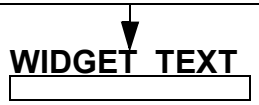
```
/* Usage */
WIDGET_DROP_DOWN_OPTION *
   widget_drop_down_option_new(
      char *string );

/* Process */
WIDGET_DROP_DOWN_OPTION *
   widget_drop_down_option_delimited_new(
      0 /* WIDGET_DROP_DOWN_LABEL_DELIMITER */,
      0 /* WIDGET_DROP_DOWN_EXTRA_DELIMITER */,
      (char *)0
      /* WIDGET_DROP_DOWN_DASH_DELIMITER */,
      0 /* not no_initial_capital */,
      (char *)0 /* sql_delimiter_string() */,
      string /* delimited_string */ );

/* Usage */
char *widget_drop_down_option_tag(
   char *widget_drop_down_option_value_string(),
   char *widget_drop_down_option_display_string() );

/* Process */
if ( !display_string || !*display_string )
{
   display_string = value_string;
}

sprintf(     static tag[],
             "\t<option value=\"%s\">%s",
             value_string,
             display_string );

/* Usage */
char *widget_drop_down_option_list_html(
   LIST* widget_drop_down_option_list,
   char *widget_drop_down_value() );

/* Process */
char html[];
char *ptr = html

if (    widget_drop_down_value
&&   *widget_drop_down_value )
{
   WIDGET_DROP_DOWN_OPTION *
      widget_drop_down_option_existing =
         widget_drop_down_option_seek(
            char *widget_drop_down_value,
            LIST *widget_drop_down_option_list );

   if ( widget_drop_down_option_existing )
   {
      /* Build a new tag. Relations are case insensitive. */
      char *widget_drop_down_option_tag(
         widget_drop_down_value /* value_string */,
         widget_drop_down_option_existing->display_string );

      ptr += sprintf( ptr,
            "%s",
            widget_drop_down_option_tag() );
   }
   else
   {
      char *widget_drop_down_option_display_string(
            WIDGET_DROP_DOWN_LABEL_DELIMITER,
            WIDGET_DROP_DOWN_DASH_DELIMITER,
            0 /* not no_initial_capital_boolean */,
            sql_delimiter_string( SQL_DELIMITER ),
            widget_drop_down_value );

      char *widget_drop_down_option_tag(
         widget_drop_down_value,
         widget_drop_down_option_display_string() );

      ptr += sprintf( ptr,
            "%s",
            widget_drop_down_option_tag() );
   }
}

for WIDGET_DROP_DOWN_OPTION *
   widget_drop_down_option in
      widget_drop_down_option_list
{
   if (    widget_drop_down_option ==
         widget_drop_down_option_existing )
            continue;

   if ( ptr != html ) ptr += sprintf( ptr, "\n" );

   ptr += sprintf( ptr,
         "%s",
         widget_drop_down_option->tag );
}

/* Usage */
char *widget_drop_down_option_static_tag(
   char *widget_operator );

/* Process */
char *widget_drop_down_option_tag(
   widget_operator /* value_string */,
   string_initial_capital(
      buffer[],
      widget_operator ) /* display_string */ );

/* Attributes */
char *value_string;
char *display_string;
char *tag;
```

WIDGET_TEXT

```
/* Usage */
WIDGET_TEXT *widget_text_new(
    char *widget_name );

/* Process */
WIDGET_TEXT *widget_text_calloc(
    void );

int prevent_carrot_boolean = 1;

/* Usage */
char *widget_text_html(
    char *widget_text->widget_name,
    int widget_text->attribute_width_max_length,
    int widget_text->widget_text_display_size,
    boolean widget_text->null_to_slash,
    boolean widget_text->prevent_carrot_boolean,
    char *on_change,
    int tab_order,
    char *value_string,
    boolean viewonly,
    char *state,
    int row_number,
    LIST *query_row_cell_list,
    char *background_color );

/* Process */
char *widget_container_name(
    widget_name,
    row_number );

char *widget_text_value_string(
    query_row_cell_list,
    value_string,
    widget_name );

char *widget_text_replace_javascript(
    char *post_change_javascript,
    int row_number,
    char *state,
    boolean null_to_slash );

boolean widget_text_autocomplete_off(
    char *widget_name );

const char *widget_text_prevent_carrot_javascript(
    boolean prevent_carrot_boolean );

char *widget_text_html_string(
    int attribute_width_max_length,
    int widget_text_display_size,
    int tab_order,
    boolean viewonly,
    char *background_color,
    char *widget_container_name(),
    char *widget_text_value_string(),
    char *widget_text_replace_javascript(),
    boolean widget_text_autocomplete_off(),
    const char *widget_text_prevent_carrot_javascript() );

/* Usage */
int widget_text_display_size(
    int WIDGET_TEXT_DEFAULT_MAX_LENGTH,
    int attribute_width );

/* Process */
if (    attribute_width >
        WIDGET_TEXT_DEFAULT_MAX_LENGTH )
{
    return WIDGET_TEXT_DEFAULT_MAX_LENGTH;
}
else
{
    return attribute_width;
}

/* Usage */
char *widget_text_value_string(
    LIST *query_row_cell_list,
    char *value_string,
    char *widget_name );

/* Process */
if ( value_string ) return value_string;

if ( list_length( query_row_cell_list ) )
{
    QUERY_CELL *query_cell_seek(
        widget_name /* attribute_name */,
        query_row_cell_list );

    if ( query_cell_seek()->display_datum )
    {
        value_string =
            query_cell_seek()->
                display_datum;
    }
    else
    {
        value_string =
            query_cell_seek()->
                select_datum;
    }
}

/* Attributes */
char *widget_name;

/* Set externally */
char *datatype_name;
int attribute_width_max_length;
int widget_text_display_size;
boolean null_to_slash;
boolean prevent_carrot_boolean;
char *on_change;
int tab_order;
char *value_string;
boolean viewonly;
```

WIDGET_PASSWORD

```
/* Usage */
WIDGET_PASSWORD *widget_password_new(
    char *widget_password_display_name() );

/* Process */
WIDGET_PASSWORD *widget_password_calloc(
    void );

WIDGET_TEXT *widget_text_new(
    widget_password_display_name
        /* widget_name */ );

/* Usage */
char *widget_password_html(
    char *widget_password->
        widget_password_display_name,
    char *widget_password->
        attribute_name,
    WIDGET_TEXT *widget_text,
    char *state,
    int row_number,
    LIST *query_row_cell_list,
    char *background_color );

/* Process */
char *widget_password_value_string(
    query_row_cell_list,
    attribute_name );

char *widget_text_html(
    widget_password_display_name /* widget_name */,
    widget_text->attribute_width_max_length,
    widget_text->widget_text_display_size,
    widget_text->null_to_slash,
    widget_text->prevent_carrot_boolean,
    widget_text->post_change_javascript,
    widget_text->tab_order,
    widget_password_value_string(),
    widget_text->viewonly,
    state,
    row_number,
    (LIST *)0 /* query_row_cell_list */,
    background_color );

/* Usage */
char *widget_password_value_string(
    LIST *query_row_cell_list,
    char *attribute_name );

/* Process */
if ( list_length( query_row_cell_list ) )
{
    QUERY_CELL *query_cell_seek(
        widget_name /* attribute_name */,
        query_row_cell_list );

    value_string =
        query_cell_seek()->
            select_datum;

    if ( *value_string )
        value_string =
            ROW_SECURITY_ASTERISK;
}

/* Usage */
char *widget_password_display_name(
    char *attribute_name );

/* Process */
snprintf(
    display_name[],
    sizeof ( display_name ),
    "%s_display",
    attribute_name );

/* Attributes */
char *widget_password_display_name;
WIDGET_TEXT *text;

/* Set externally */
char *attribute_name;
```



WIDGET_NON_EDIT_TEXT

```
/* Usage */
WIDGET_NON_EDIT_TEXT *
    widget_non_edit_text_new(
        char *widget_name );

/* Process */
WIDGET_NON_EDIT_TEXT *
    widget_non_edit_text_calloc(
        void );

/* Usage */
char *widget_non_edit_text_html(
    LIST *query_row_cell_list,
    char *widget_name,
    boolean no_initial_capital_boolean );

/* Process */
if ( query_row_cell_list )
{
    char *widget_container_value(
        ATTRIBUTE_MULTI_KEY_DELIMITER,
        query_row_cell_list,
        widget_name );

    if (    widget_container_value()
    &&  boolean string_exists_character(
            widget_name,
            ATTRIBUTE_MULTI_KEY_DELIMITER ) )
    {
        widget_container_value =
            widget_drop_down_option_display_string(
                WIDGET_DROP_DOWN_LABEL_DELIMITER,
                WIDGET_DROP_DOWN_DASH_DELIMITER,
                no_initial_capital_boolean,
                sql_delimiter_string(
                    ATTRIBUTE_MULTI_KEY_DELIMITER ),
                widget_container_value );
    }
}

char *widget_non_edit_text_html_string(
    widget_name,
    no_initial_capital_boolean,
    widget_container_value() );

/* Usage */
char *widget_non_edit_text_html_string(
    char *widget_name,
    boolean no_initial_capital_boolean,
    widget_container_value() );

/* Process */

/* Attributes */
char *widget_name;

/* Set externally */
boolean no_initial_capital_boolean;
```

WIDGET_ENCRYPT

```
/* Usage */
WIDGET_ENCRYPT *widget_encrypt_new(
    char *widget_name );

/* Process */
WIDGET_ENCRYPT *widget_encrypt_calloc(
    void );

/* Usage */
char *widget_encrypt_html(
    int WIDGET_TEXT_DEFAULT_DISPLAY_SIZE,
    int row_number,
    LIST *query_row_cell_list,
    char *widget_name,
    int attribute_width_max_length,
    int tab_order );
```

```
/* Process */
char *widget_encrypt_compare_name(
    const char *WIDGET_ENCRYPT_COMPARE_LABEL,
    char *widget_name,
    int row_number );

char *widget_container_key(
    widget_name,
    row_number );

if ( query_row_cell_list )
{
    char *widget_container_value(
        (char)0 /* ATTRIBUTE_MULTI_KEY_DELIMITER */,
        query_row_cell_list,
        widget_name );
}
```

```
char *widget_encrypt_html_string(
    int attribute_width_max_length,
    int widget_text_display_size,
    int tab_order,
    char *widget_encrypt_compare_name(),
    char *widget_container_key(),
    char *widget_container_value(),

/* Attributes */
char *widget_name;

/* Set externally */
int attribute_width_max_length;
int tab_order;
```

WIDGET_CHECKBOX (1)

```
/* Usage */
WIDGET_CHECKBOX *widget_checkbox_new(
    char *widget_name );

/* Process */
WIDGET_CHECKBOX *widget_checkbox_calloc(
    void );

/* Usage */
char *widget_checkbox_submit_javascript(
    char *form_name,
    char *recall_save_javascript );

/* Process */
char javascript[];
char *ptr = javascript;

if ( save_javascript )
{
    ptr += sprintf( ptr,
        "%s && ",
        recall_save_javascript );
}

sprintf( ptr,
    "push_button_submit( '%s' )",
    form_name );

/* Usage */
boolean widget_checkbox_dictionary_checked(
    char *widget_name,
    DICTIONARY *dictionary );

/* Process */
char *dictionary_get(
    widget_name,
    dictionary );

if ( !dictionary_get() )
    return 0;
else
if ( tolower( *dictionary_get() ) == 'y' )
    return 1;
else
    return 0;

/* Usage */
char *widget_checkbox_html(
    char *widget_name,
    char *prompt,
    char *onclick,
    int tab_order,
    char *image_source,
    char *state,
    int row_number,
    LIST *query_row_cell_list,
    char *background_color );

/* Process */
if ( !widget_name && !prompt ) exit( 1 );

if ( widget_name )
{
    char *widget_container_key(
        widget_name,
        row_number );
}

if ( widget_name && list_length( query_row_cell_list ) )
{
    boolean widget_checkbox_checked(
        LIST *query_row_cell_list,
        char *widget_name )
}

char *javascript_replace(
    onclick /* post_change_javascript */,
    state,
    row_number );

char *widget_checkbox_html_string(
    prompt,
    tab_order,
    image_source,
    background_color,
    widget_container_key(),
    widget_checkbox_checked(),
    javascript_replace() );
```

WIDGET_CHECKBOX (2)

```
/* Usage */
char *widget_checkbox_html_string(
    char *prompt,
    int tab_order,
    char *image_source,
    char *background_color,
    char *widget_container_key(),
    boolean widget_checkbox_checked(),
    char *javascript_replace() );

/* Process */
char html_string[];
char *ptr = html_string;

if ( !prompt && !widget_container_key ) exit( 1 );

if ( image_source && *image_source )
{
    ptr += sprintf( ptr,
        "<image src=\"%s\">",
        image_source );
}

if ( prompt )
{
    string_initial_capital( display[], prompt );
}
else
{
    string_initial_capital(
        display[],
        widget_container_key );
}

ptr += sprintf( ptr, "%s", display );

ptr += sprintf( ptr,
    "<input type=checkbox value=y" );

if ( widget_container_key )
{
    ptr += sprintf( ptr,
        " name=\"%s\" id=\"%s\"",
        widget_container_key,
        widget_container_key );
}

if ( javascript_replace && *javascript_replace )
{
    ptr += sprintf( ptr,
        " onClick=\"%s\"",
        javascript_replace );
}

if ( tab_order > 0 )
{
    static char *widget_container_tab_order_html(
        tab_order );

    ptr += sprintf( ptr,
        "%s",
        widget_container_tab_order_html() );
}

if ( widget_checkbox_checked )
{
    ptr += sprintf( ptr, " checked" );
}

if ( background_color && *background_color )
{
    char *widget_container_background_color_html(
        background_color );

    ptr += sprintf( ptr,
        "%s",
        widget_container_background_color_html() );
}

sprintf( "ptr, ">" );

/* Attributes */
char *widget_name;

/* Set externally */
char *prompt;
char *onclick;
int tab_order;
char *image_source;
```

WIDGET_RADIO

```
/* Usage */
WIDGET_RADIO *widget_radio_new(
    char *widget_name );

/* Process */
WIDGET_RADIO *widget_radio_calloc( void );

/* Usage */
char *widget_radio_html(
    char *widget_name,
    char *value,
    char *prompt,
    char *onclick );

/* Process */
/* Attributes */
char *widget_name;

/* Set externally */
char *value;
char *prompt;
char *onclick;
```

WIDGET_NOTEPAD

```
/* Usage */
WIDGET_NOTEPAD *widget_notepad_new(
    char *widget_name );

/* Process */
WIDGET_NOTEPAD *widget_notepad_calloc(
    void );

/* Usage */
char *widget_notepad_html(
    char *widget_name,
    int attribute_size,
    boolean null_to_slash,
    char *post_change_javascript,
    int tab_order,
    char *state,
    int row_number,
    LIST *query_row_cell_list,
    char *background_color );

/* Process */
char *widget_container_key(
    widget_name,
    row_number );

if ( list_length( query_row_cell_list ) )
{
    char *widget_container_value(
        (char)0 /* ATTRIBUTE_MULTI_KEY_DELIMITER */,
        query_row_cell_list,
        widget_name );
}

char *widget_text_replace_javascript(
    post_change_javascript,
    row_number,
    state,
    null_to_slash );

char *widget_notepad_html_string(
    int tab_order,
    char *background_color,
    char *widget_container_key(),
    char *widget_container_value(),
    int attribute_size,
    int WIDGET_NOTEPAD_COLUMNS,
    int WIDGET_NOTEPAD_ROWS,
    char *widget_text_replace_javascript() );

/* Attributes */
char *widget_name;

/* Set externally */
int attribute_size;
boolean null_to_slash;
char *post_change_javascript;
int tab_order;
```

WIDGET_HIDDEN

```
/* Usage */
WIDGET_HIDDEN *widget_hidden_new(
    char *widget_name );

/* Process */
WIDGET_HIDDEN *widget_hidden_calloc(
    void );

/* Usage */
char *widget_hidden_html(
    int row_number,
    LIST *query_row_cell_list,
    WIDGET_HIDDEN *widget_hidden );

/* Process */
char *widget_hidden_container_key(
    widget_name,
    row_number );

char *widget_hidden_value(
    query_row_cell_list,
    widget_hidden->widget_name,
    widget_hidden->value_string );

return
char *widget_hidden_html_string(
    char *widget_hidden_container_key(),
    char *widget_hidden_value() );

/* Usage */
char *widget_hidden_value(
    LIST *query_row_cell_list,
    char *widget_name,
    char *value_string );

/* Process */
if ( value_string ) return value_string;

char *widget_container_value(
    ATTRIBUTE_MULTI_KEY_DELIMITER,
    query_row_cell_list,
    widget_name );

/* Usage */
char *widget_hidden_container_key(
    char *widget_name,
    int row_number );

/* Process */
if ( row_number )
{
    return
    char *widget_container_key(
        widget_name,
        row_number );
}
else
{
    return widget_name;
}

/* Attributes */
char *widget_name;

/* Set externally */
char *value_string;
```

WIDGET_UPLOAD (1)

```
/* Usage */
WIDGET_UPLOAD *widget_upload_new(
    char *widget_name );

/* Process */
WIDGET_UPLOAD *widget_upload_calloc(
    void );

char *widget_upload_recall_widget_name(
    WIDGET_UPLOAD_RECALL_PREFIX,
    widget_name );

/* Usage */
char *widget_upload_prompt_frame_html_string(
    char *widget_name,
    char *recall_widget_name,
    int tab_order );

/* Process */
char html_string[];
char *ptr = html_string;

static char *widget_upload_post_change_javascript(
    recall_widget_name );

ptr += sprintf( ptr,
    "<input name=%s id=%s %s %s %s onChange=\"%s\"",
    widget_name,
    widget_name,
    "type=file",
    "accept=\"*\"",
    "value=\"\"",
    widget_upload_post_change_javascript() );

if ( tab_order > 0 )
{
    static char *widget_container_tab_order_html(
        tab_order );

    ptr += sprintf( ptr,
        "%s",
        widget_container_tab_order_html() );
}

ptr += sprintf( ptr, ">" );

char *widget_text_html_string(
    1024 /* attribute_width_max_length */,
    20 /* widget_text_display_size */,
    0 /* tab_order */,
    1 /* viewonly */,
    (char *)0 /* background_color */,
    recall_widget_name /* widget_container_name */,
    (char *)0 /* widget_text_value_string */,
    (char *)0 /* widget_text_replace_javascript */,
    1 /* widget_text_autocomplete_off */,
    widget_text_prevent_carrot_javascript(
        0 /* not prevent_carrot_boolean */ ) );

ptr += sprintf( ptr,
    "%s",
    widget_text_html_string() );

free( widget_text_html_string() );

/* Usage */
static char *widget_upload_post_change_javascript(
    char *widget_upload_recall_widget_name );

/* Process */
snprintf(
    static char javascript[],
    sizeof ( javascript ),
    "form_cookie_upload_copy( this, '%s' )",
    widget_upload_recall_widget_name );

/* Usage */
char *widget_upload_recall_widget_name(
    const char *WIDGET_UPLOAD_RECALL_PREFIX,
    char *widget_name );

/* Process */

/* Attributes */
char *widget_name;
char *recall_widget_name;

/* Set externally */
char *application_name;
char *session_key;
int tab_order;
char *post_change_javascript;
```

WIDGET_UPLOAD (2)

```
/* Usage */
char *widget_upload_edit_frame_html(
    char *widget_upload->application_name,
    char *widget_upload->session_key,
    char *widget_upload->widget_name,
    LIST *query_row_cell_list,
    char *background_color );

/* Process */
static char *appaserver_parameter_upload_directory();

static char *appaserver_parameter_document_root();

char *widget_container_value(
    (char)0 /* ATTRIBUTE_MULTI_KEY_DELIMITER */,
    query_row_cell_list,
    widget_name );

if (    !widget_container_value()
||      !*widget_container_value() )
{
    return NULL;
}

char *widget_upload_source_filespecification(
    char *application_name,
    char *appaserver_parameter_upload_directory(),
    char *widget_container_value() );

boolean file_exists_boolean(
    widget_upload_source_filespecification() );

if ( !file_exists_boolean() )
{
    return
    widget_non_edit_text_html_string(
        widget_name,
        1 /* no_initial_capital_boolean */,
        widget_container_value() );
}

char *widget_upload_filename_basename(
    char *widget_container_value() );

char *widget_upload_filename_extension(
    char *widget_container_value() );

char *widget_upload_hypertext(
    application_name,
    session_key,
    widget_upload_filename_basename(),
    widget_upload_filename_extension() );

char *widget_upload_link_system_string(
    char *appaserver_parameter_document_root(),
    char *widget_upload_source_filespecification(),
    char *widget_upload_hypertext() );

system( widget_upload_link_system_string() );

char *widget_upload_edit_frame_html_string(
    char *background_color,
    char *widget_container_value(),
    char *widget_upload_hypertext() );

/* Usage */
char *widget_upload_hypertext(
    char *application_name,
    char *session_key,
    char *widget_upload_filename_basename(),
    char *widget_upload_filename_extension() );

/* Process */
if ( !widget_upload_filename_extension )
{
    snprintf(
        static hypertext[],
        sizeof ( hypertext ),
        "/appaserver_data/%s/%s_%s",
        application_name,
        widget_upload_filename_basename,
        session_key );
}
else
{
    snprintf(
        static hypertext[],
        sizeof ( hypertext ),
        "/appaserver_data/%s/%s_%s.%s",
        application_name,
        widget_upload_filename_basename,
        session_key,
        widget_upload_filename_extension );
}
```

WIDGET_YES_NO (1)

```
/* Usage */
WIDGET_YES_NO *widget_yes_no_new(
    char *widget_name );

/* Process */
WIDGET_YES_NO *widget_yes_no_calloc(
    void );

/* Usage */
char *widget_yes_no_html(
    char *widget_name,
    char *post_change_javascript,
    int tab_order,
    char *state,
    int row_number,
    LIST *query_row_cell_list,
    char *background_color );

/* Process */
char *widget_container_key(
    widget_name,
    row_number );

char *widget_container_value(
    (char)0 /* ATTRIBUTE_MULTI_KEY_DELIMITER */,
    query_cell_list,
    widget_name );

boolean widget_yes_no_affirmative_boolean(
    char *widget_container_value() );

boolean widget_yes_no_not_affirmative_boolean(
    char *widget_container_value() );

char *javascript_replace(
    post_change_javascript,
    state,
    row_number );

char *widget_yes_no_html_string(
    widget_container_key(),
    widget_yes_no_affirmative_boolean(),
    widget_yes_no_not_affirmative_boolean(),
    javascript_replace(),
    tab_order,
    background_color );

/* Attributes */
char *widget_name;

/* Set externally */
char *post_change_javascript;
int tab_order;
```

WIDGET_YES_NO (2)

```
/* Usage */
char *widget_yes_no_html_string(
    char *widget_container_key(),
    boolean widget_yes_no_affirmative_boolean(),
    boolean widget_yes_no_not_affirmative_boolean(),
    char *javascript_replace(),
    int tab_order,
    char *background_color );

/* Process */
char html_string[];
char *ptr = html_string;

ptr += sprintf( ptr,
    "<select name=\"%s\" id=\"%s\"",
    widget_container_key,
    widget_container_key );

if ( tab_order > 0 )
{
    ptr += sprintf( ptr,
        "%s",
        static char *widget_container_tab_order_html(
            tab_order ) );
}

if (    javascript_replace
&&   *javascript_replace )
{
    ptr += sprintf( ptr,
        "%s",
        widget_container_javascript_html(
            javascript_replace ) );
}

if (    background_color
&&   *background_color )
{
    ptr += sprintf( ptr,
        "%s",
        widget_container_background_color_html(
            background_color ) );
}

ptr += sprintf( ptr, ">" );

LIST *option_list = list_new();

if ( widget_yes_no_affirmative_boolean )
{
    WIDGET_YES_NO_OPTION *
        widget_yes_no_option =
            widget_yes_no_option_new(
                1 /* yes_boolean */,
                0 /* not no_boolean */,
                0 /* not select_boolean */ );

    list_set(
        option_list,
        widget_yes_no_option->
            widget_drop_down_option );

    WIDGET_YES_NO_OPTION *
        widget_yes_no_option =
            widget_yes_no_option_new(
                0 /* not yes_boolean */,
                1 /* no_boolean */,
                0 /* not select_boolean */ );

    list_set(
        option_list,
        widget_yes_no_option->
            widget_drop_down_option );
}
else
if ( widget_yes_no_not_affirmative_boolean )
{
    WIDGET_YES_NO_OPTION *
        widget_yes_no_option =
            widget_yes_no_option_new(
                0 /* not yes_boolean */,
                1 /* no_boolean */,
                0 /* not select_boolean */ );

    list_set(
        option_list,
        widget_yes_no_option->
            widget_drop_down_option );

    WIDGET_YES_NO_OPTION *
        widget_yes_no_option =
            widget_yes_no_option_new(
                1 /* yes_boolean */,
                0 /* not no_boolean */,
                0 /* not select_boolean */ );

    list_set(
        option_list,
        widget_yes_no_option->
            widget_drop_down_option );
}
```

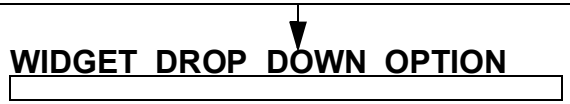
WIDGET_YES_NO (3)

```
/* widget_yes_no_html_string() continued */
else
{
    WIDGET_YES_NO_OPTION *
        widget_yes_no_option =
            widget_yes_no_option_new(
                0 /* not yes_boolean */,
                0 /* not no_boolean */,
                1 /* select_boolean */ );

    list_set(
        option_list,
        widget_yes_no_option->
            widget_drop_down_option );

    WIDGET_YES_NO_OPTION *
        widget_yes_no_option =
            widget_yes_no_option_new(
                1 /* yes_boolean */,
                0 /* not no_boolean */,
                0 /* not select_boolean */ );

    list_set(
        option_list,
        widget_yes_no_option->
            widget_drop_down_option );

    WIDGET_YES_NO_OPTION *
        widget_yes_no_option =
            widget_yes_no_option_new(
                0 /* not yes_boolean */,
                1 /* no_boolean */,
                0 /* not select_boolean */ );

    list_set(
        option_list,
        widget_yes_no_option->
            widget_drop_down_option );

}
```

```
char *widget_drop_down_option_list_html(
    option_list,
    (char *)0 /* widget_drop_down_value */ );

ptr += sprintf( ptr,
    "%s\n",
    widget_drop_down_option_list_html() );

free( widget_drop_down_option_list_html() );

sprintf( ptr,
    "%s",
    widget_drop_down_select_close_tag(
        0 /* not null_boolean*/,
        0 /* not not_null_boolean */,
        0 /* not bottom_select_boolean */ ) );
```

WIDGET_YES_NO_OPTION

```
/* Usage */
WIDGET_YES_NO_OPTION *widget_yes_no_option_new(
    boolean yes_boolean,
    boolean no_boolean,
    boolean select_boolean );

/* Process */
WIDGET_YES_NO_OPTION *
    widget_yes_no_option_calloc(
        void );

WIDGET_DROP_DOWN_OPTION *
    widget_drop_down_option =
        widget_drop_down_option_calloc();

if ( yes_boolean )
{
    widget_drop_down_option->value_string = "y";
    widget_drop_down_option->display_string = "Yes";
}
else
if ( no_boolean )
{
    widget_drop_down_option->value_string = "n";
    widget_drop_down_option->display_string = "No";
}
else
if ( select_boolean )
{
    widget_drop_down_option->value_string =
        WIDGET_SELECT_OPERATOR;

    widget_drop_down_option->display_string =
        strdup(
            string_initial_capital(
                display[],
                WIDGET_SELECT_OPERATOR ) );
}
else
{
    exit( 1 );
}

widget_drop_down_option->tag =
    strdup(
        widget_drop_down_option_tag(
            widget_drop_down_option->
                value_string,
            widget_drop_down_option->
                display_string ) );

/* Attribute */
WIDGET_DROP_DOWN_OPTION *
    widget_drop_down_option;
```



WIDGET_DATE (1)

```
/* Usage */
WIDGET_DATE *widget_date_new(
    char *widget_name );

/* Process */
WIDGET_DATE *widget_date_calloc( void );

widget_date_calloc()->validate_date_boolean = 1;

/* Usage */
char *widget_date_html(
    char *widget_name,
    char *datatype_name,
    int attribute_width_max_length,
    int display_size,
    boolean null_to_slash_boolean,
    boolean validate_date_boolean,
    char *post_change_javascript,
    int tab_order,
    char *application_name,
    char *login_name,
    char *state,
    int row_number,
    LIST *query_row_cell_list,
    char *background_color );

/* Process */
char *widget_container_key(
    widget_name,
    row_number );

boolean attribute_is_current_date_time(
    datatype_name );

boolean attribute_is_current_date(
    datatype_name );

if (    (   attribute_is_current_date_time()
||          attribute_is_current_date() )
&&  string_strcmp(
            state,
            APPASERVER_INSERT_STATE ) == 0 )
{
    char *widget_container_value() =
        widget_date_login_name_now_date_string(
            attribute_width_max_length,
            application_name,
            login_name );
}
else
{
    char *widget_container_value(
        (char)0 /* ATTRIBUTE_MULTI_KEY_DELIMITER */,
        query_row_cell_list,
        widget_name );
}

char *widget_text_javascript_replace_on_change(
    post_change_javascript,
    row_number,
    state,
    null_to_slash_boolean );

boolean widget_date_calendar_boolean(
    datatype_name,
    state,
    widget_container_value() );

if ( widget_date_calendar_boolean() )
{
    char *appaserver_user_date_format(
        char *application_name,
        char *login_name );

    char *widget_date_format_template(
        char *appaserver_user_date_format() );

    char *widget_date_calendar_image_tag(
        char *widget_container_key(),
        char *widget_date_format_template() );
}

char *widget_date_html_string(
    int attribute_width_max_length,
    int display_size,
    char *widget_container_key(),
    char *widget_container_value(),
    boolean validate_date_boolean,
    char *widget_text_javascript_replace_on_change(),
    char *widget_date_calendar_image_tag(),
    int tab_order,
    char *background_color );

/* Usage */
boolean widget_date_calendar_boolean(
    char *datatype_name,
    char *state,
    char *widget_container_value );

/* Process */
boolean attribute_is_current_date_time(
    datatype_name );

if (    attribute_is_current_date_time()
&&  string_strcmp(
            state,
            APPASERVER_INSERT_STATE ) == 0 )
{
    return 0;
}

if (    ( attribute_is_date_time()
||      attribute_is_current_date_time() )
&&  widget_container_value
&&  *widget_container_value )
{
    return 0;
}

return 1;

/* Attributes */
char *widget_name;
boolean validate_date_boolean;

/* Set externally */
char *datatype_name;
char *application_name;
char *login_name;
int attribute_width_max_length;
int display_size;
boolean null_to_slash_boolean;
char *post_change_javascript;
int tab_order;
```

WIDGET_DATE (2)

```
/* Usage */
char *widget_date_login_name_now_date_string(
    int attribute_width_max_length,
    char *application_name,
    char *login_name );

/* Process */
if ( attribute_width_max_length == 10 )
{
    char *date_now_string =
        date_now_yyyy_mm_dd(
            date_utc_offset() );
}
else
if ( attribute_width_max_length == 16 )
{
    char *date_now_string =
        date_now16(
            date_utc_offset() );
}
else
if ( attribute_width_max_length == 19 )
{
    char *date_now_string =
        date_now19(
            date_utc_offset() );
}
else
    exit( 1 );

static enum date_convert_format_enum
    login_name_enum =
        date_convert_unknown;

if ( login_name_enum == date_convert_unknown )
{
    login_name_enum =
        date_convert_login_name_enum(
            application_name,
            login_name );
}

if ( login_name_enum == date_convert_international )
{
    return date_now_string;
}

char *widget_date_portion_date(
    char *date_now_string );

char *widget_date_portion_time(
    char *date_now_string );

char *date_convert_source_international(
    login_name_enum,
    widget_date_portion_date()
        /* source_date_string */ );

char *widget_date_combine_date_string(
    char *date_convert_source_international(),
    char *widget_date_portion_time() );
```

WIDGET_TIME

```
/* Usage */
WIDGET_TIME *widget_time_new(
    char *widget_name );

/* Process */
WIDGET_TIME *widget_time_calloc( void );

/* Usage */
char *widget_time_html(
    char *widget_name,
    int attribute_width_max_length,
    int widget_text_display_size,
    boolean null_to_slash,
    char *post_change_javascript,
    int tab_order,
    char *state,
    int row_number,
    LIST *query_row_cell_list,
    char *background_color );

/* Process */
char *widget_container_key(
    widget_name,
    row_number );

char *widget_container_value(
    (char)0 /* ATTRIBUTE_MULTI_KEY_DELIMITER */,
    query_row_cell_list,
    widget_name );

char *widget_text_replace_javascript(
    post_change_javascript,
    row_number,
    state,
    null_to_slash );

char *widget_time_html_string(
    int attribute_width_max_length,
    int widget_text_display_size,
    int tab_order,
    char *background_color,
    char *widget_container_key(),
    char *widget_container_value(),
    char *widget_text_replace_javascript() );

/* Attributes */
char *widget_name;
int attribute_width_max_length;
int widget_text_display_size;
boolean null_to_slash;
char *post_change_javascript;
int tab_order;
```

WIDGET_ANCHOR

```
/* Usage */
WIDGET_ANCHOR *widget_anchor_calloc(
    void );

/* Process */

/* Usage */
char *widget_anchor_html(
    char *hypertext_reference,
    char *target_frame,
    char *prompt );

/* Process */

/* Attributes */
char *hypertext_reference;
char *target_frame;
char *prompt;
```

WIDGET_TABLE_OPEN

```
/* Usage */
WIDGET_TABLE_OPEN *widget_table_open_new(
    boolean border_boolean );

/* Process */
WIDGET_TABLE_OPEN *widget_table_open_calloc(
    void );
```

```
/* Process */

/* Usage */
char *widget_table_open_tag(
    boolean border_boolean,
    int cell_spacing,
    int cell_padding );
```

```
/* Process */

/* Attributes */
boolean border_boolean;

/* Set externally */
int cell_spacing;
int cell_padding;
```

WIDGET_TABLE_HEADING

```
/* Usage */
WIDGET_TABLE_HEADING *widget_table_heading_new(
    char *heading_string );

/* Process */
WIDGET_TABLE_HEADING
```

```
*widget_table_heading_calloc( void );

/* Usage */
char *widget_table_heading_tag(
    char *heading_string );
```

```
/* Process */

/* Attribute */
char *heading_string;
```

WIDGET_LINE_BREAK

```
/* Usage */
WIDGET_LINE_BREAK *widget_line_break_new(\
    void );

/* Process */
```

```
WIDGET_LINE_BREAK *widget_line_break =
    WIDGET_LINE_BREAK *widget_line_break_calloc(
    void );

widget_line_break->tag = “<br>”;
```

```
/* Attribute */
char *tag;
```

WIDGET_TABLE_ROW

```
/* Usage */
WIDGET_TABLE_ROW *widget_table_row_calloc(
    void );

/* Process */

/* Usage */
char *widget_table_row_tag(
    char *background_color );

/* Process */
```

WIDGET_TABLE_DATA

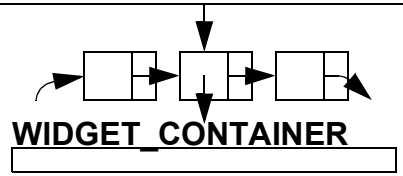
```
/* Usage */
WIDGET_TABLE_DATA *widget_table_data_calloc(
    void );

/* Process */

/* Usage */
char *widget_table_data_tag(
    boolean align_right,
    int column_span,
    int width_em,
    char *style );

/* Process */

/* Attributes */
boolean align_right;
int column_span;
int width_em;
char *style;
```

WIDGET_TABLE_CLOSE

```
/* Usage */
WIDGET_TABLE_CLOSE *widget_table_close_calloc(
    void );

/* Process */

/* Usage */
char *widget_table_close_tag(
    void );

/* Process */
```

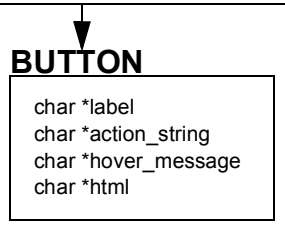
WIDGET_AJAX

```
/* Usage */
WIDGET_AJAX *widget_ajax_new(
    LIST *ajax_client_widget_container_list );

/* Process */
WIDGET_AJAX *widget_ajax_calloc(
    void );

/* Usage */
char *widget_ajax_html(
    LIST *ajax_client_widget_container_list,
    int row_number,
    LIST *query_row_cell_list,
    char *background_color );

/* Process */
char html[];
char *ptr = html;

for WIDGET_CONTAINER *widget_container in
    ajax_client_widget_container_list
{
    char *widget_container_html(
        (char *)0 /* state */,
        row_number,
        query_row_cell_list,
        (RELATION_JOIN_ROW *)0 /* relation_join_row */,
        background_color,
        widget_container );

    ptr += sprintf( ptr,
        "%s\n",
        widget_container_html() );

    free( widget_container_html() );
}

/* Attribute */
LIST *ajax_client_widget_container_list;
```



WIDGET_BUTTON

```
/* Usage */
WIDGET_BUTTON *widget_button_new(
    char *label );

/* Process */
WIDGET_BUTTON *widget_button_calloc(
    void );

widget_button_calloc()->button =
    button_new(
        label );

/* Usage */
LIST *widget_button_submit_reset_container_list(
    char *form_name );

/* Process */
LIST *widget_container_list = list_new();

list_set(
    widget_container_list,
    ( WIDGET_CONTAINER *widget_container =
        widget_container_new(
            button, (char *)0 ) ) );

widget_container->widget_button->button =
    button_submit(
        form_name,
        (char *)0 /* form_multi_select_all_javascript */,
        (char *)0 /* recall_save_javascript */,
        (char *)0 /* form_verify_notepad_widths */ );

list_set(
    widget_container_list,
    ( WIDGET_CONTAINER *widget_container =
        widget_container_new(
            button, (char *)0 ) ) );

widget_container->widget_button->button =
    button_reset(
        form_name,
        (char *)0 /* javascript_replace */ );

list_set(
    widget_container_list,
    ( WIDGET_CONTAINER *widget_container =
        widget_container_new(
            button, (char *)0 ) ) );

widget_container->widget_button->button =
    button_back();

list_set(
    widget_container_list,
    ( WIDGET_CONTAINER *widget_container =
        widget_container_new(
            button, (char *)0 ) ) );

widget_container->widget_button->button =
    button_forward();

/* Usage */
char *widget_button_html(
    int row_number,
    WIDGET_BUTTON *widget_button );

/* Process */
if ( widget_button->button->html )
{
    return widget_button->button->html;
}

char *button_html(
    widget_button->button->label,
    widget_button->button->action_string,
    widget_button->button->hover_title,
    row_number );

/* Attribute */
BUTTON *button;
```



BUTTON (1)

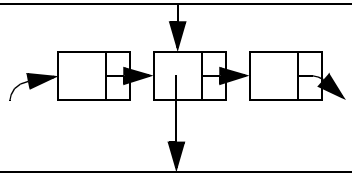
```
/* Usage */
BUTTON *button_new(
    char *label );

/* Process */
BUTTON *button_calloc(
    void );

/* Usage */
BUTTON *button_submit(
    char *form_name,
    char *form_multi_select_all_javascript(),
    char *recall_save_javascript(),
    char *form_verify_notepad_widths_javascript() );

/* Process */
BUTTON *button = button_calloc();

char *button->html =
    button_submit_html(
        char *form_name,
        const char *BUTTON_SUBMIT_DISPLAY,
        char *form_multi_select_all_javascript,
        char *recall_save_javascript,
        char *form_verify_notepad_widths_javascript,
        const char *BUTTON_SUBMIT_MESSAGE );

/* Usage */
BUTTON *button_reset(
    char *form_name,
    char *javascript_replace() );

/* Process */
BUTTON *button = button_calloc();

char *button->html =
    button_reset_html(
        char *form_name,
        const char
            WIDGET_MULTI_LEFT_RIGHT_DELIMITER,
        char *javascript_replace,
        const char *BUTTON_RESET_MESSAGE );

/* Usage */
BUTTON *button_back( void );

/* Process */
BUTTON *button = button_calloc();

char *button->html =
    button_input_html(
        "Back" /* display */,
        "history.back()" /* action_string */,
        BUTTON_BACK_MESSAGE );

/* Usage */
BUTTON *button_forward( void );

/* Process */
BUTTON *button = button_calloc();

char *button->html =
    button_input_html(
        "Forward" /* display */,
        "timlib_history_forward()" /* action_string */,
        BUTTON_FORWARD_MESSAGE );

/* Usage */
BUTTON *button_restart_drillthru(
    char *post_choose_folder_action_string() );

/* Process */
BUTTON *button = button_calloc();

char *button->html =
    button_anchor_html(
        "Restart Drillthru" /* display */,
        post_choose_folder_action_string
            /* href_action_string */,
        (char *)0 /* onclick_action_string */,
        BUTTON_DRILLTHRU_MESSAGE
            /* hover_message */ );

/* Usage */
BUTTON *button_to_top( void );

/* Process */
BUTTON *button = button_calloc();

button->html =
    button_to_top_html(
        char *BUTTON_IMAGE_RELATIVE_DIRECTORY,
        char *BUTTON_TO_TOP_MESSAGE );

/* Usage */
BUTTON *button_recall(
    char *recall_load_javascript() );

/* Process */
BUTTON *button = button_calloc();

button->html =
    button_input_html(
        "Recall" /* display */,
        recall_load_javascript /* action_string */,
        BUTTON_RECALL_MESSAGE );

/* Usage */
BUTTON *button_drillthru_skip( void );

/* Process */
BUTTON *button = button_calloc();

button->html =
    button_drillthru_skip_html(
        char *BUTTON_SKIP_MESSAGE,
        char WIDGET_MULTI_LEFT_RIGHT_DELIMITER );

/* Usage */
BUTTON *button_help(
    char *application_name,
    char *html_help_file_anchor );

/* Process */
BUTTON *button = button_calloc();

button->html =
    button_help_html(
        char *application_name,
        char *html_help_file_anchor );
```

BUTTON (2)

```
/* Usage */
char *button_html(
    char *label,
    char *action_string,
    char *hover_message,
    int row_number );

/* Process */
char html[];
char *ptr = html;

char *javascript_replace(
    action_string /* post_change_javascript */,
    (char *)0 /* state */,
    row_number );

ptr += sprintf( ptr,
    "<button type=button onClick=\"%s\"",
    javascript_replace() );

if ( hover_message )
{
    ptr += sprintf( ptr,
        " title=\"%s\"",
        hover_message );
}

char *button_display(
    char *label );

ptr += sprintf( ptr,
    ">%s</button>",
    button_display() );

/* Usage */
char *button_input_html(
    char *display,
    char *action_string,
    char *hover_message );

/* Process */
char html[];
char *ptr = html;

ptr += sprintf( ptr,
    "<input type=button value=\"%s\"",
    display );

ptr += sprintf( ptr,
    " onClick=\"%s\"",
    action_string );

if ( hover_message )
{
    ptr += sprintf( ptr,
        " title=\"%s\"",
        hover_message );
}

ptr += sprintf( ptr, ">" );

/* Usage */
char *button_display(
    char *label );

/* Process */
char *string_initial_capital( label, label );

/* Usage */
char *button_anchor_html(
    char *display,
    char *href_action_string,
    char *anchor_action_string,
    char *hover_message );

/* Process */
char html[];
char *ptr = html;

ptr += sprintf( ptr,
    "<button type=button" );

if ( onclick_action_string )
{
    ptr += sprintf( ptr,
        " onClick=\"%s\"",
        onclick_action_string );
}

if ( hover_message )
{
    ptr += sprintf( ptr,
        " title=\"%s\"",
        hover_message );
}

ptr += sprintf( ptr,
    " <a href=\"%s\">%s</a></button>",
    href_action_string,
    display );

/* Public */
char *button_list_html(
    LIST *button_list );

void button_list_free(
    LIST *button_list );

/* Attributes */
char *label;

/* Set externally */
char *action_string;
char *hover_message;
char *html;
```

RADIO_LIST

```
/* Usage */
RADIO_LIST *radio_list_new(
    char *FORM_RADIO_LIST_NAME,
    LIST *radio_pair_list,
    char *initial_value );

/* Process */
RADIO_LIST *radio_list_calloc( void );

LIST *radio_value_list = list_new();

for RADIO_PAIR *radio_pair in radio_pair_list
{
    list_set(
        radio_value_list,
        radio_value_new(
            form_radio_list_name,
            radio_pair->radio_value
                /* radio_value_string */,
            radio_pair->radio_display,
            initial_value ) );
}

char *radio_list_html(
    LIST *radio_value_list );

/* Attributes */
LIST *radio_value_list;
char *html;
```



RADIO_VALUE

```
/* Usage */
RADIO_VALUE *radio_value_new(
    char *radio_list_name,
    char *radio_pair->radio_value
        /* radio_value_string */,
    char *radio_pair->radio_display,
    char *initial_value );

/* Process */
RADIO_VALUE *radio_value_calloc( void );

char *radio_value_html(
    char *radio_list_name,
    char *radio_value_string,
    char *radio_display,
    char *initial_value );

/* Attribute */
char *html;
```

RADIO_PAIR

```
/* Usage */
RADIO_PAIR *radio_pair_new(
    char *radio_value,
    char *radio_display );

/* Process */
RADIO_PAIR *radio_pair_calloc( void );

/* Attributes */
char *radio_value;
char *radio_display;
```

POST_CHOOSE_FOLDER (1)

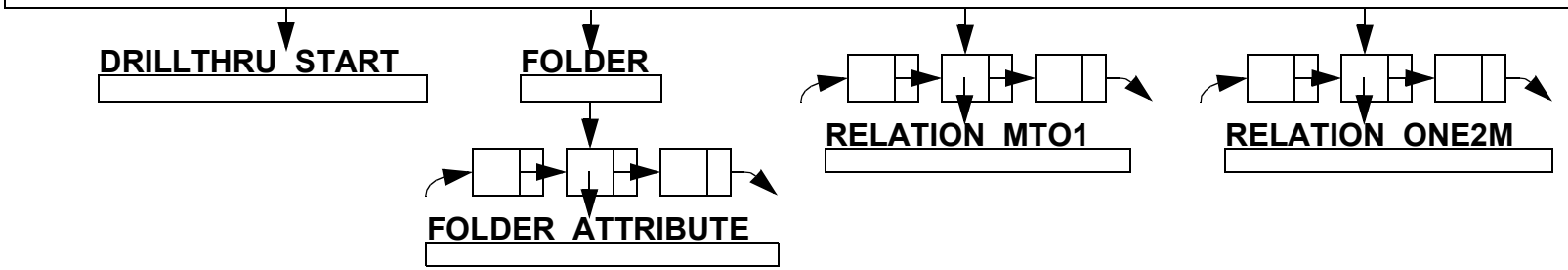
```
/* Usage */
POST_CHOOSE_FOLDER *post_choose_folder_new(
    /* See session_folder_integrity_exit() */
    char *session_folder->application_name,
    char *session_folder->session->session_key,
    char *session_folder->login_name,
    char *session_folder->role_name,
    char *session_folder->folder_name,
    char *session_folder->state,
    char *appaserver_parameter->data_directory
         /* appaserver_data_directory */ );

/* Process */
POST_CHOOSE_FOLDER *post_choose_folder_calloc(
    void );

DRILLTHRU_START *drillthru_start =
    drillthru_start_new(
        application_name,
        session_key,
        login_name,
        role_name,
        folder_name,
        state,
        appaserver_data_directory  );

if ( drillthru_start->participating_boolean )
        return this;

FOLDER *folder =
    folder_fetch(
        folder_name,
        role_name,
        (LIST *)0 /* exclude_attribute_name_list */,
        1 /* fetch_folder_attribute_list */,
        0 /* not fetch_attribute */ );

boolean post_choose_folder_fetch_relation_mto1_isa(
    const char *APPASERVER_INSERT_STATE,
    char *state );

if ( post_choose_folder_fetch_relation_mto1_isa() )
{
    LIST *relation_mto1_isa_list(
        (LIST *)0 /* mto1_isa_list Pass in null */,
        folder_name /* many_folder_name */,
        folder->folder_attribute_primary_key_list,
        0 /* not fetch_relation_one2m_list */,
        0 /* not fetch_relation_mto1_list */ );
}

LIST *relation_one2m_pair_list(
    folder_name /* one_folder_name */,
    folder->folder_attribute_primary_key_list );

char *post_choose_folder_form_name(
    const char *FORM_APPASERVER_PROMPT,
    const char *FORM_APPASERVER_TABLE,
    int list_length( relation_one2m_pair_list() )
        /* relation_one2m_pair_list_length */,
    char *folder->appaserver_form );

boolean post_choose_folder_isa_drop_down_boolean(
    int list_length( relation_mto1_isa_list() )
        /* relation_mto1_isa_list_length */ );

boolean post_choose_folder_prompt_insert_boolean(
    const char *APPASERVER_INSERT_STATE,
    const char *FORM_APPASERVER_PROMPT,
    char *state,
    char *post_choose_folder_form_name() );

boolean post_choose_folder_table_insert_boolean(
    const char *APPASERVER_INSERT_STATE,
    const char *FORM_APPASERVER_TABLE,
    char *state,
    char *post_choose_folder_form_name() );

boolean post_choose_folder_prompt_lookup_boolean(
    /* Not this */
    const char *APPASERVER_INSERT_STATE,
    const char *FORM_APPASERVER_PROMPT,
    char *state,
    char *post_choose_folder_form_name() );

boolean post_choose_folder_table_edit_boolean(
    /* Not this */
    const char *APPASERVER_INSERT_STATE,
    const char *FORM_APPASERVER_TABLE,
    char *state,
    char *post_choose_folder_form_name() );

char *appaserver_error_filename(
    application_name );

char *post_choose_folder_system_string(
    application_name,
    session_key,
    login_name,
    role_name,
    folder_name,
    post_choose_folder_isa_drop_down_boolean(),
    post_choose_folder_prompt_insert_boolean(),
    post_choose_folder_table_insert_boolean(),
    post_choose_folder_prompt_lookup_boolean(),
    post_choose_folder_table_edit_boolean(),
    appaserver_error_filename(),
    list_first( relation_mto1_isa_list() )
        /* relation_mto1_isa_first */ );
```

POST_CHOOSE_FOLDER (2)

```
/* Usage */
char *post_choose_folder_system_string(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    boolean post_choose_folder_isa_drop_down_boolean,
    boolean post_choose_folder_prompt_insert_boolean,
    boolean post_choose_folder_table_insert_boolean,
    boolean
        post_choose_folder_prompt_lookup_boolean,
    boolean post_choose_folder_table_edit(),
    char *appaserver_error_filename(),
    RELATION_MTO1 *
        list_first( relation_mto1_isa_list() )
            /* relation_mto1_isa_first */ );

/* Process */
if ( post_choose_folder_isa_drop_down_boolean )
{
    char *system_string =
        char *execute_system_string_folder_isa(
            CHOOSE_ISA_EXECUTABLE,
            session_key,
            login_name,
            role_name,
            folder_name /* many_folder_name */,
            relation_mto1_isa_first->one_folder_name,
            appaserver_error_filename );
}
else
if ( post_choose_folder_prompt_insert_boolean )
{
    char *system_string =
        char *execute_system_string_prompt_insert(
            PROMPT_INSERT_EXECUTABLE,
            session_key,
            login_name,
            role_name,
            folder_name,
            (char *)0 /* dictionary_separate_send_string */,
            appaserver_error_filename );
}
else
if ( post_choose_folder_table_insert_boolean )
{
    char *system_string =
        char *execute_system_string_table_insert(
            TABLE_INSERT_EXECUTABLE,
            session_key,
            login_name,
            role_name,
            folder_name,
            FRAMESET_PROMPT_FRAME
                /* target_frame */,
            (char *)0 /* results_string */,
            (char *)0 /* error_string */,
            (char *)0 /* dictionary_separate_send_string */,
            appaserver_error_filename );
}
else
if ( post_choose_folder_prompt_lookup_boolean )
{
    char *system_string =
        char *execute_system_string_prompt_lookup(
            PROMPT_LOOKUP_EXECUTABLE,
            session_key,
            login_name,
            role_name,
            folder_name,
            (char *)0 /* dictionary_separate_send_string */,
            appaserver_error_filename );
}
else
if ( post_choose_folder_table_edit_boolean )
{
    char *system_string =
        char *execute_system_string_table_edit(
            TABLE_EDIT_EXECUTABLE,
            session_key,
            login_name,
            role_name,
            folder_name,
            FRAMESET_PROMPT_FRAME /* target_frame */,
            (char *)0 /* sub_sub_title */,
            (char *)0 /* dictionary_separate_send_string */,
            appaserver_error_filename );
}
```

POST_CHOOSE_FOLDER (3)

```
/* Usage */
char *post_choose_folder_action_string(
    char *POST_CHOOSE_FOLDER_EXECUTABLE,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *state,
    char *folder_name );

/* Process */
char *appaserver_parameter_cgi_directory();

char *application_server_address();

char *application_ssl_support_boolean(
    application_name );

char *application_http_prompt(
    appaserver_parameter_cgi_directory(),
    appliction_server_address(),
    application_ssl_support_boolean() );

sprintf(
    static action_string[],
    "%s/%s?%s+%s+%s+%s+%s+%s",
    application_http_prompt(),
    POST_CHOOSE_FOLDER_EXECUTABLE,
    application_name,
    session_key,
    login_name,
    role_name,
    state );

if ( folder_name )
{
    sprintf(
        action_string + strlen( action_string ),
        "+%s",
        folder_name );
}

/* Usage */
char *post_choose_folder_anchor_tag(
    char *target_frame,
    char *post_choose_folder_href_string() );

/* Process */
sprintf(
    static anchor_tag[],
    "<a href=\"%s\" target=%s>",
    post_choose_folder_href_string,
    target_frame );

/* Attributes */
DRILLTHRU_START *drillthru_start;
FOLDER *folder;
boolean fetch_relation_mto1_isa;
LIST *relation_mto1_isa_list;
LIST *relation_one2m_pair_list;
char *form_name;
boolean isa_drop_down;
boolean prompt_insert_boolean;
boolean table_insert_boolean;
boolean prompt_lookup_boolean;
boolean table_edit_boolean;
char *appaserver_error_filename;
char *system_string;

/* Driver */
if ( post_choose_folder->
       drillthru_start->
       participating_boolean )
{
    document_content_type_output();

    printf(
        "%s\n",
        post_choose_folder->
            drillthru_start->
            drillthru_document->
            document_form_html );
}
else
{
    system( system_string );
}
```

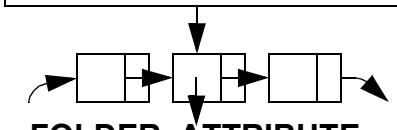


POST_DRILLTHRU (1)

```
/* Usage */
POST_DRILLTHRU *post_drillthru_new(
    /* See session_folder_integrity_exit() */
    char *session_folder->session->application_name,
    char *session_folder->session->session_key,
    char *session_folder->session->login_name,
    char *session_folder->role_name,
    char *session_folder->folder
        /* current_folder_name */,
    char *session_folder->state,
    char *appaserver_parameter->
        appaserver_data_directory,
    DICTIONARY *post_dictionary->
        original_post_dictionary );

/* Process */
POST_DRILLTHRU *post_drillthru_calloc(
    void );

char *post_drillthru_base_folder_name(
    char *DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
    char *DRILLTHRU_BASE_KEY,
    DICTIONARY *original_post_dictionary );

LIST *folder_attribute_list(
    post_drillthru_base_folder_name(),
    (LIST *)0 /* exclude_attribute_name_list */,
    1 /* fetch_attribute */,
    1 /* cache_boolean */ );

LIST *folder_attribute_date_name_list(
    folder_attribute_list() );

DICTIONARY_SEPARATE_DRILLTHRU *
    dictionary_separate_drillthru =
        dictionary_separate_drillthru_new(
            original_post_dictionary,
            application_name,
            login_name,
            folder_attribute_date_name_list(),
            folder_attribute_list() );

dictionary_append_dictionary(
    dictionary_separate_drillthru->
        drillthru_dictionary,
    dictionary_separate_drillthru->
        non_prefixed_dictionary );

DRILLTHRU_CONTINUE *drillthru_continue =
    drillthru_continue_new(
        dictionary_separate_drillthru->
            drillthru_dictionary /* in/out */,
        application_name,
        session_key,
        login_name,
        role_name,
        current_folder_name,
        state,
        appaserver_data_directory );

/* Drillthru participating */
if ( drillthru_continue->drillthru_document )
        return this;

/* Drillthru skipped or completed */
DICTIONARY *dictionary_separate_send_dictionary(
    (DICTIONARY *)0 /* sort_dictionary */,
    DICTIONARY_SEPARATE_SORT_PREFIX,
    (DICTIONARY *)0 /* query_dictionary */,
    DICTIONARY_SEPARATE_QUERY_PREFIX,
    dictionary_separate_drillthru->drillthru_dictionary,
    DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
    (DICTIONARY *)0 /* ignore_dictionary */,
    DICTIONARY_SEPARATE_IGNORE_PREFIX,
    (DICTIONARY *)0 /* no_display_dictionary */,
    DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
    (DICTIONARY *)0 /* pair_dictionary */,
    DICTIONARY_SEPARATE_PAIR_PREFIX,
    (DICTIONARY *)0 /* non_prefixed_dictionary */ );

char *dictionary_separate_send_string(
    dictionary_separate_send_dictionary() );

char *appaserver_error_filename(
    application_name );

if ( strcmp( state, APPASERVER_INSERT_STATE ) == 0 )
{
    char *system_string =
    execute_system_string_prompt_insert(
        PROMPT_INSERT_EXECUTABLE,
        session_key,
        login_name,
        role_name,
        post_drillthru_base_folder_name(),
        dictionary_separate_send_string(),
        appaserver_error_filename );
}
else
{
    char *system_string =
    execute_system_string_prompt_lookup(
        PROMPT_LOOKUP_EXECUTABLE,
        session_key,
        login_name,
        role_name,
        post_drillthru_base_folder_name(),
        dictionary_separate_send_string(),
        appaserver_error_filename() );
}
```

POST_DRILLTHRU (2)

```
/* Attributes */
char *base_folder_name;
LIST *folder_attribute_list;
LIST *folder_attribute_date_name_list;
DICTIONARY_SEPARATE_DRILLTHRU *
   dictionary_separate_drillthru;
DRILLTHRU_CONTINUE *drillthru_continue;
DICTIONARY *dictionary_separate_send_dictionary;
char *dictionary_separate_send_string;
char *appaserver_error_filename;
char *system_string;

/* Driver */
if ( post_drillthru->
       drillthru_continue->
       drillthru_document )
{
   printf("%s\n",
                        post_drillthru->
                           drillthru_continue->
                           drillthru_document->
                           document_form_html );
}
else
{
   system( post_drillthru->system_string );
}
```



FOLDER_ATTRIBUTE

ATTRIBUTE *attribute

DICTIONARY_SEPARATE_DRILLTHRU

DICTIONARY *query_dictionary
DICTIONARY *drillthru_dictionary
DICTIONARY *non_prefixed_dictionary

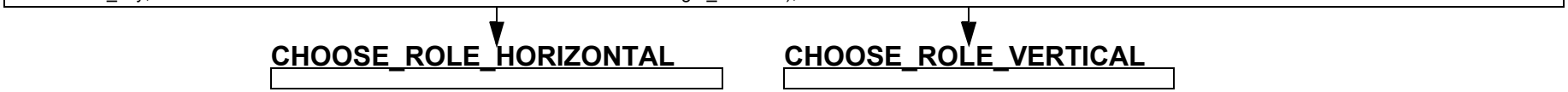
DRILLTHRU_CONTINUE

DRILLTHRU_DOCUMENT *
drillthru_document

DICTIONARY

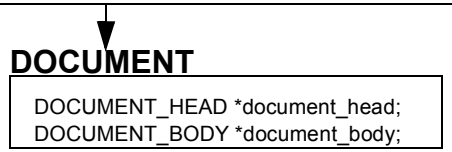
CHOOSE_ROLE

```
/* Usage */
CHOOSE_ROLE *choose_role_new(
    char *application_name,
    char *session_key,
    char *login_name,
    boolean application_menu_horizontal_boolean );

/* Process */
CHOOSE_ROLE *choose_role_calloc(
    void );

char *application_title_string( application_name );

char *choose_role_title_string(
    char *application_title_string(),
    char *login_name,
    boolean application_menu_horizontal_boolean );

LIST *appaserver_user_role_name_list(
    login_name );

if ( !list_length( appaserver_user_role_name_list() )
        exit( 1 );

char *post_choose_role_action_string(
    POST_CHOOSE_ROLE_EXECUTABLE,
    application_name,
    session_key,
    login_name );

if ( application_menu_horizontal_boolean )
{
    CHOOSE_ROLE_HORIZONTAL *
        choose_role_horizontal =
            choose_role_horizontal_new(
                application_name,
                login_name,
                application_title_string(),
                choose_role_title_string(),
                appaserver_user_role_name_list(),
                post_choose_role_action_string(),
                FRAMESET_PROMPT_FRAME
                    /* target_frame */ );
}
else
{
    CHOOSE_ROLE_VERTICAL *
        choose_role_vertical =
            choose_role_vertical_new(
                application_name,
                application_title_string(),
                choose_role_title_string(),
                appaserver_user_role_name_list(),
                post_choose_role_action_string(),
                FRAMESET_MENU_FRAME
                    /* target_frame */ );
}

/* Driver */
if ( choose_role->choose_role_horizontal )
{
    printf(   "%s\n",
              choose_role->
                  choose_role_horizontal->
                  document_form_html );
}
else
if ( choose_role->choose_role_vertical )
{
    printf(   "%s\n",
              choose_role->
                  choose_role_vertical->
                  document_form_html );
}

/* Attributes */
char *application_title_string;
char *title_string;
LIST *appaserver_user_role_name_list;
char *post_choose_role_action_string;
CHOOSE_ROLE_HORIZONTAL *choose_role_horizontal;
CHOOSE_ROLE_VERTICAL *choose_role_vertical;
```



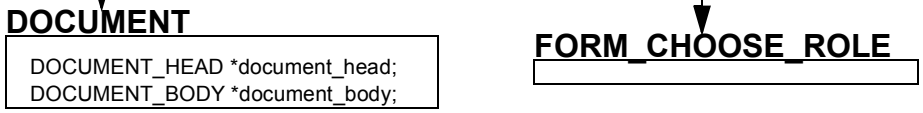
CHOOSE_ROLE_HORIZONTAL

```
/* Usage */
CHOOSE_ROLE_HORIZONTAL *
    choose_role_horizontal_new(
        char *application_name,
        char *login_name,
        char *application_title_string(),
        char *choose_role_title_string(),
        LIST *appaserver_user_role_name_list(),
        char *post_choose_role_action_string(),
        char *target_frame
            /* FRAMESET_PROMPT_FRAME */ );

/* Process */
CHOOSE_ROLE_HORIZONTAL *
    choose_role_horizontal_calloc(
        void );

char *choose_role_horizontal_menu_html(
    login_name,
    appaserver_user_role_name_list,
    target_frame,
    post_choose_role_action_string );

DOCUMENT *document =
    document_new(
        application_name,
        application_title_string,
        choose_role_title_string,
        (char *)0 /* sub_title_string */,
        (char *)0 /* sub_sub_title_string */,
        (char *)0 /* notepad */,
        (char *)0 /* onload_javascript_string */,
        choose_role_horizontal_menu_html(),
        document_head_menu_setup_string(
            1 /* menu_boolean */ ),
        (char *)0 /* calendar_setup_string */,
        javascript_include_string(
            application_name,
            (LIST *)0 /* javascript_filename_list */ ) );

char *document_form_html(
    document->html,
    document->document_head->html,
    document_head_close_tag(),
    document->document_body->html,
    (char *)0 /* form_html */,
    document_body_close_tag(),
    document_close_tag() );

/* Usage */
char *choose_role_horizontal_menu_html(
    char *login_name,
    LIST *appaserver_user_role_name_list,
    char *target_frame,
    char *post_choose_role_action_string );

/* Process */
char html[];
char *ptr = html;

ptr += sprintf( ptr,
    "%s\n",
    MENU_UNORDERED_LIST_TAG );

char *menu_item_heading_span_tag(
    MENU_ITEM_HORIZONTAL_CLASS_NAME,
    "Role" /* heading_string */ );

ptr += sprintf( ptr,
    "%s\n",
    menu_item_heading_span_tag() );

ptr += sprintf( ptr, "\t<ul>\n" );

for char *role_name in appaserver_user_role_name_list
{
    char *menu_item_span_tag(
        MENU_ITEM_HORIZONTAL_CLASS_NAME,
        role_name /* item */ );

    char *menu_item_href_string(
        role_name /* item */,
        post_choose_role_action_string() );

    char *menu_item_anchor_tag(
        target_frame,
        menu_item_href_string() );

    char *menu_item_horizontal_html(
        menu_item_span_tag(),
        menu_item_anchor_tag() );

    ptr += sprintf( ptr,
        "%s\n",
        menu_item_horizontal_html() );

    free( menu_item_horizontal_html() );
}

ptr += sprintf( ptr, "\t</ul>\n" );

char *string_initial_capital(
    buffer[],
    login_name );

ptr += sprintf( ptr,
    "<li><a><label style=\"color:black\">" );

ptr += sprintf( ptr,
    "%s</label></a>\n",
    string_initial_capital() );

sprintf( ptr, "</ul>" );

/* Attributes */
char *menu_html;
DOCUMENT *document;
char *document_form_html;
```



CHOOSE_ROLE_VERTICAL

```
/* Usage */
CHOOSE_ROLE_VERTICAL *
   choose_role_vertical_new(
      char *application_name,
      char *application_title_string(),
      char *choose_role_title_string(),
      LIST *appaserver_user_role_name_list(),
      char *post_choose_role_action_string(),
      char *target_frame
         /* FRAMESET_MENU_FRAME */ );

/* Process */
CHOOSE_ROLE_VERTICAL *
   choose_role_vertical_calloc(
      void );

DOCUMENT *document =
   document_new(
      application_name,
      application_title_string(),
      choose_role_title_string(),
      (char *)0 /* sub_title_string */,
      (char *)0 /* sub_sub_title_string */,
      (char *)0 /* notepad */,
      (char *)0 /* onload_javascript_string */,
      (char *)0 /* menu_html */,
      (char *)0 /* menu_setup_string */,
      (char *)0 /* calendar_setup_string */,
      javascript_include_string(
         application_name,
         (LIST *)0 /* javascript_filename_list */ ) );

FORM_CHOOSE_ROLE *form_choose_role =
   form_choose_role_new(
      appaserver_user_role_name_list(),
      post_choose_role_action_string(),
      target_frame,
      FORM_CHOOSE_ROLE_NAME,
      CHOOSE_ROLE_DROP_DOWN_NAME );

char *document_form_html(
   document->html,
   document->document_head->html,
   document_head_close_tag(),
   document->document_body->html,
   form_choose_role->html,
   document_body_close_tag(),
   document_close_tag() );

/* Attributes */
DOCUMENT *document;
FORM_CHOOSE_ROLE *form_choose_role;
char *document_form_html;
```



POST_CHOOSE_ROLE

```
/* Usage */
POST_CHOOSE_ROLE *post_choose_role_new(
    int argc,
    char **argv,
    char *CHOOSE_ROLE_DROP_DOWN_NAME );

/* Process */
POST_CHOOSE_ROLE *post_choose_role_calloc(
    void );

char *sql_injection_escape_application_name =
    security_sql_injection_escape(
        SECRUITY_ESCAPE_CHARACTER_STRING,
        argv[ 1 ] );

char *sql_injection_escape_session_key =
    security_sql_injection_escape(
        SECRUITY_ESCAPE_CHARACTER_STRING,
        argv[ 2 ] );

char *sql_injection_escape_login_name =
    security_sql_injection_escape(
        SECRUITY_ESCAPE_CHARACTER_STRING,
        argv[ 3 ] );

char *post_choose_role_name(
    argc,
    argv[ 4 ],
    CHOOSE_ROLE_DROP_DOWN_WIDGET_NAME );

void session_environment_set(
    sql_injection_escape_application_name );

boolean application_menu_horizontal_boolean(
    sql_injection_escape_application_name );

char *execute_system_string_menu(
    MENU_EXECUTABLE,
    sql_injection_escape_application_name,
    sql_injection_escape_session_key,
    sql_injection_escape_login_name,
    post_choose_role_name(),
    application_menu_horizontal_boolean() );

/* Usage */
char *post_choose_role_name(
    int argc,
    char *argv_4,
    char *CHOOSE_ROLE_DROP_DOWN_WIDGET_NAME );

/* Process */
if ( argc == 5 )
{
    return
    security_sql_injection_escape(
        SECRUITY_ESCAPE_CHARACTER_STRING,
        argv_4 );
}

POST_DICTIONARY *post_dictionary =
    post_dictionary_stdin_new(
        (char *)0 /* WIDGET_UPLOAD_RECALL_PREFIX */,
        (char *)0 /* application_name */,
        (char *)0 /* upload_directory */,
        (LIST *)0 /* upload_filename_list */ );

return
char *security_sql_injection_escape(
    dictionary_get(
        CHOOSE_ROLE_DROP_DOWN_WIDGET_NAME,
        post_dictionary->original_post_dictionary ) );

/* Usage */
char *post_choose_role_action_string(
    char *POST_CHOOSE_ROLE_EXECUTABLE,
    char *application_name,
    char *session_key,
    char *login_name );

/* Process */
static action_string[] = {0};

if ( *action_string ) return action_string;

char *appaserver_parameter_cgi_directory();

char *application_server_address();

char *application_ssl_support_boolean(
    application_name );

char *application_http_prompt(
    appaserver_parameter_cgi_directory(),
    appliction_server_address(),
    application_ssl_support_boolean() );

sprintf(
    static action_string[],
    "%s/%s?%s+%s+%s",
    application_http_prompt(),
    POST_CHOOSE_ROLE_EXECUTABLE,
    application_name,
    session_key,
    login_name );

/* Driver */
system(
    post_choose_role->
        execute_system_string_menu );

/* Attributes */
char *sql_injection_escape_application_name;
char *sql_injection_escape_session_key;
char *sql_injection_escape_login_name;
char *name;
char *execute_system_string_menu;
```

CHOOSE_ISA (1)

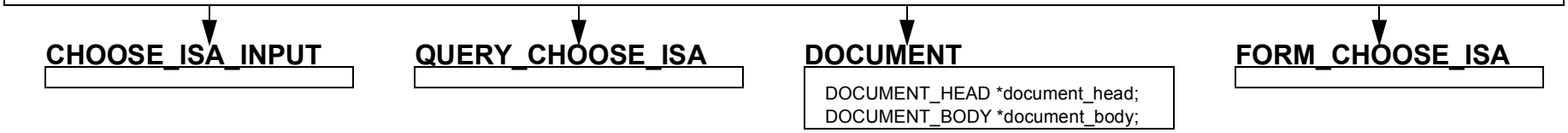
```
/* Usage */
CHOOSE_ISA *choose_isa_new(
    char *session_folder->
        session->
        application_name,
    char *session_folder->
        session->
        session_key,
    char *session_folder->
        session->
        login_name,
    char *session_folder->role_name,
    char *session_folder->folder_name
        /* insert_folder_name */,
    char *security_sql_injection_escape(
        one_folder_name ),
    boolean application_menu_horizontal_boolean() );

/* Process */
CHOOSE_ISA *choose_isa_calloc(
    void );

CHOOSE_ISA_INPUT *choose_isa_input =
    choose_isa_input_new(
        application_name,
        session_key,
        login_name,
        role_name,
        one_folder_name,
        application_menu_horizontal_boolean );

if ( choose_isa_input->populate_drop_down_process )
{
    char *choose_isa_command_line(
        choose_isa_input->
            populate_drop_down_process->
            command_line
                /* process_command_line */,
        session_key,
        login_name,
        role_name,
        insert_folder_name,
        choose_isa_input->security_entity_where,
        appaserver_error_filename(
            application_name );

    delimited_list =
        list_pipe_fetch(
            choose_isa_command_line() );
}
else
{
    QUERY_CHOOSE_ISA *query_choose_isa =
        query_choose_isa_new(
            application_name,
            login_name,
            one_folder_name,
            choose_isa_input->
                one_folder->
                folder_attribute_primary_list,
            choose_isa_input->
                security_entity_where );

    delimited_list =
        query_choose_isa->
            query_row_delimited_list;
}

char *choose_isa_title_string(
    char *insert_folder_name );

char *choose_isa_sub_sub_title_string(
    char *insert_folder_name,
    char *one_folder_name );
```

CHOOSE_ISA (2)

```
/* Process (continued) */
DOCUMENT *document =
    document_new(
        application_name,
        application_title_string( application_name ),
        choose_isa_title_string(),
        (char *)0 /* sub_title_string */,
        choose_isa_sub_sub_title_string(),
        (char *)0 /* notepad */,
        (char *)0 /* onload_javascript_string */,
        (choose_isa_input->menu) ?
            choose_isa_input->menu->html : (char *)0,
        document_head_menu_setup_string(
            (choose_isa_input->menu)
                ? 1 : 0 /* menu_boolean */ ),
        (char *)0 /* calendar_setup_string */,
        javascript_include_string(
            application_name,
            (LIST *)0 /* javascript_filename_list */ ) );

char *choose_isa_prompt_message(
    char *WIDGET_DROP_DOWN_DASH_DELIMITER,
    LIST *choose_isa_input->
        one_folder->
        folder_attribute_primary_key_list );

char *choose_isa_action_string(
    const char *POST_CHOOSE_ISA_EXECUTABLE,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *insert_folder_name,
    char *one_folder_name );

FORM_CHOOSE_ISA *form_choose_isa =
    form_choose_isa_new(
        WIDGET_LOOKUP_CHECKBOX_NAME,
        choose_isa_prompt_message(),
        choose_isa_input->
            one_folder->
            folder_attribute_primary_key_list,
        delimited_list,
        choose_isa_input->one_folder->no_initial_capital,
        choose_isa_action_string(),
        FRAMESET_PROMPT_FRAME /* target_frame */ );

char *document_form_html(
    document->html,
    document->document_head->html,
    document_head_close_tag(),
    document->document_body->html,
    form_choose_isa->html,
    document_body_close_tag(),
    document_close_tag() );

free( form_choose_isa->html );

/* Usage */
char *choose_isa_command_line(
    char *process_command_line,
    char *session_key,
    char *login_name,
    char *role_name,
    char *insert_folder_name,
    char *security_entity_where(),
    char *appaserver_error_filename() );

/* Process */
char *query_drop_down_process_command_line(
    (char *)0 /* application_name */,
    session_key,
    login_name,
    role_name,
    (char *)0 /* state */,
    insert_folder_name /* many_folder_name */,
    (char *)0 /* one_folder_name */,
    (char *)0 /* related_column */,
    (char *)0 /* populate_drop_down_process_name */,
    (DICTIONARY *)0 /* dictionary */,
    security_entity_where /* where_string */,
    process_command_line,
    appaserver_error_filename );
```

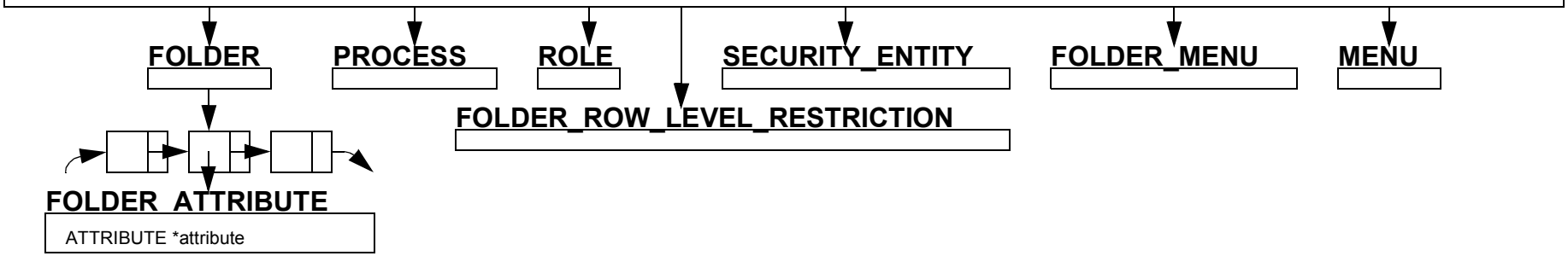
CHOOSE_ISA (3)

```
/* Attributes */
CHOOSE_ISA_INPUT *choose_isa_input;
QUERY_CHOOSE_ISA *query_choose_isa;
LIST *delimited_list;
char *title_string;
char *sub_sub_title_string;
char *prompt_message;
char *action_string;
DOCUMENT *document;
FORM_CHOOSE_ISA *form_choose_isa;
char *document_form_html;

/* Driver */
printf( "%s\n", document_form_html );
```



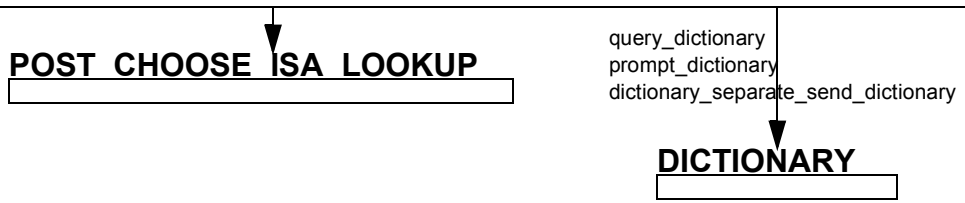
CHOOSE_ISA_INPUT

```
/* Usage */
CHOOSE_ISA_INPUT *choose_isa_input_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *one_folder_name,
    boolean application_menu_horizontal_boolean );

/* Process */
CHOOSE_ISA_INPUT *
    choose_isa_input_calloc(
        void );

FOLDER *one_folder =
    folder_fetch(
        one_folder_name,
        (LIST *)0 /* exclude_attribute_name_list */,
        /* Sets folder_attribute_primary_list */
        /* Sets folder_attribute_primary_key_list */
        /* Sets folder_attribute_name_list */
        1 /* fetch_folder_attribute_list */,
        1 /* fetch_attribute */,
        0 /* not cache_boolean */ );

if ( one_folder->populate_drop_down_process_name )
{
    PROCESS *populate_drop_down_process =
        process_fetch(
            one_folder->
                populate_drop_down_process_name,
            (char *)0 /* document_root */,
            (char *)0 /* relative_source_directory */,
            1 /* check_executable_inside_filesystem */,
            appaserver_parameter_mount_point() );
}

ROLE *role =
    role_fetch(
        role_name,
        0 /* not fetch_role_attribute_exclude_list */ );

FOLDER_ROW_LEVEL_RESTRICTION *
    folder_row_level_restriction =
        folder_row_level_restriction_fetch(
            one_folder_name );

SECURITY_ENTITY *security_entity =
    security_entity_new(
        login_name,
        one_folder_name,
        folder_row_level_restriction->
            non_owner_forbid,
        role->override_row_restrictions );

if ( application_menu_horizontal_boolean )
{
    FOLDER_MENU *folder_menu =
        folder_menu_fetch(
            application_name,
            session_key,
            role_name,
            appaserver_parameter_data_directory(),
            0 /* not folder_menu_remove_boolean */ );

    MENU *menu =
        menu_new(
            application_name,
            session_key,
            login_name,
            role_name,
            1 /* application_menu_horizontal_boolean */,
            folder_menu->count_list );
}

/* Attributes */
FOLDER *one_folder;
PROCESS *populate_drop_down_process;
ROLE *role;
FOLDER_ROW_LEVEL_RESTRICTION *
    folder_row_level_restriction;
SECURITY_ENTITY *security_entity;
FOLDER_MENU *folder_menu;
MENU *menu;
```



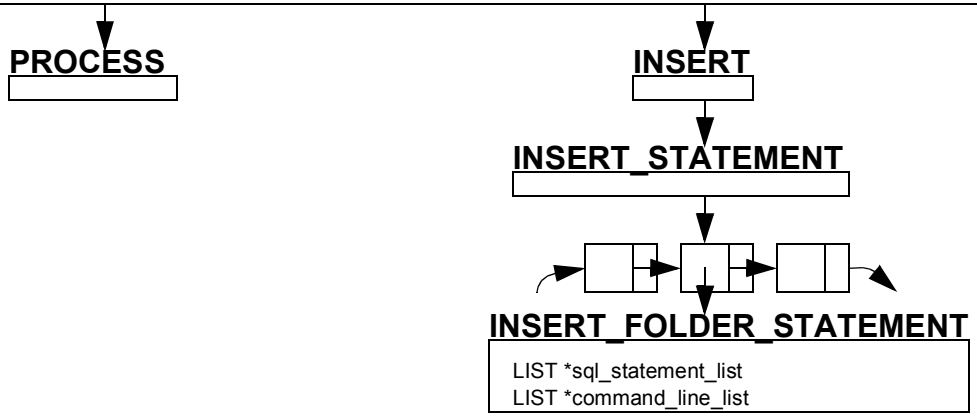
POST_CHOOSE_ISA (1)

```
/* Usage */
POST_CHOOSE_ISA *post_choose_isa_new(
    /* See session_folder_integrity_exit() */
    char *session_folder->application_name,
    char *session_folder->session_key,
    char *session_folder->login_name,
    char *session_folder->role_name,
    char *session_folder->folder_name,
    char *security_sql_injection_escape(
        one_folder_name ) );

/* Process */
POST_CHOOSE_ISA *post_choose_isa_calloc(
    void );

POST_CHOOSE_ISA_INPUT *
    post_choose_isa_input =
        post_choose_isa_input_new(
            application_name,
            login_name,
            role_name,
            folder_name,
            one_folder_name );

LIST *post_choose_isa_primary_data_list(
    post_choose_isa_input->
        one_folder->
        folder_attribute_primary_key_list,
    post_choose_isa->
        post_choose_isa_input->
        dictionary_separate->
        prompt_dictionary );

int post_choose_isa_primary_data_list_length(
    LIST *post_choose_isa_primary_data_list() );

if ( !post_choose_isa_primary_data_list_length() )
{
    char *post_choose_isa_skip_system_string(
        TABLE_INSERT_EXECUTABLE,
        PROMPT_INSERT_EXECUTABLE,
        session_key,
        login_name,
        role_name,
        folder_name,
        post_choose_isa_input->
            folder->
            appaserver_form,
        post_choose_isa_input->
            appaserver_error_filespecification );

    return this;
}

boolean post_choose_isa_lookup_checkbox_checked(
    WIDGET_LOOKUP_CHECKBOX_NAME,
    post_choose_isa_input->
        dictionary_separate->
        non_prefixed_dictionary );

if ( post_choose_isa_lookup_checkbox_checked() )
{
    POST_CHOOSE_ISA_LOOKUP *
        post_choose_isa_lookup_new(
            session_key,
            login_name,
            role_name,
            post_choose_isa_input->one_folder,
            post_choose_isa_input->
                appaserver_error_filename,
            post_choose_isa_primary_data_list() );

    execute_system_string_table_edit =
        post_choose_isa_lookup_new()->
            execute_system_string_table_edit;

    return this;
}

DICTIONARY *post_choose_isa_query_dictionary(
    QUERY_RELATION_OPERATOR_PREFIX,
    QUERY_EQUAL,
    post_choose_isa_input->
        one_folder->
        folder_attribute_primary_key_list,
    post_choose_isa_primary_data_list() );

DICTIONARY *post_choose_isa_prompt_dictionary(
    post_choose_isa_input->
        folder->
        folder_attribute_primary_key_list,
    post_choose_isa_primary_data_list() );

DICTIONARY *dictionary_separate_send_dictionary(
    (DICTIONARY *)0 /* sort_dictionary */,
    DICTIONARY_SEPARATE_SORT_PREFIX,
    post_choose_isa_query_dictionary(),
    DICTIONARY_SEPARATE_QUERY_PREFIX,
    (DICTIONARY *)0 /* drillthru_dictionary */,
    DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
    (DICTIONARY *)0 /* ignore_dictionary */,
    DICTIONARY_SEPARATE_IGNORE_PREFIX,
    (DICTIONARY *)0 /* no_display_dictionary */,
    DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
    (DICTIONARY *)0 /* pair_dictionary */,
    DICTIONARY_SEPARATE_PAIR_PREFIX,
    post_choose_isa_prompt_dictionary()
        /* non_prefixed_dictionary */ );

char *dictionary_separate_send_string(
    dictionary_separate_send_dictionary() );
```



POST_CHOOSE_ISA (2)

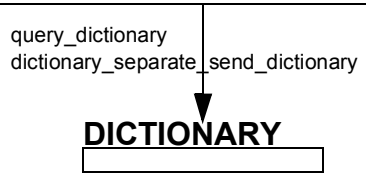
```
/* Process (continued) */
if ( post_choose_isa_input->
       folder->
       post_change_process_name )
{
    PROCESS *post_change_process =
       process_fetch(
          post_choose_isa_input->
             folder->
             post_change_process_name,
          (char *)0 /* document_root */,
          (char *)0 /* relative_source */,
          1 /* check_executable */,
          appaserver_parameter_mount_point() );
}

INSERT *insert_new(
    application_name,
    session_key,
    login_name,
    role_name,
    folder_name,
    post_choose_isa_input->
       folder->
       folder_attribute_primary_key_list
       /* root_primary_key_list */,
    post_choose_insert_input->
       folder->
       folder_attribute_list
       /* folder_attribute_append_isa_list */,
    (LIST *)0 /* relation_mto1_isa_list */,
    post_choose_isa_prompt_dictionary()
       /* in/out */,
    (DICTIONARY *)0 /* multi_row_dictionary */,
    (LIST *)0 /* ignore_name_list */,
    post_change_process );

    /* Attributes */
    POST_CHOOSE_ISA_INPUT *post_choose_isa_input;
    LIST *primary_data_list;
    int primary_data_list_length;
    char *skip_system_string;
    boolean lookup_checkbox_checked;
    POST_CHOOSE_ISA_LOOKUP *post_choose_isa_lookup;
    DICTIONARY *query_dictionary;
    DICTIONARY *prompt_dictionary;
    DICTIONARY *dictionary_separate_send_dictionary;
    char *dictionary_separate_send_string;
    PROCESS *post_change_process;
    INSERT *insert;

    /* Set externally */
    char *execute_system_string_table_edit;
```



```
/* Usage */
LIST *post_choose_isa_primary_data_list(
    LIST *one_folder->folder_attribute_primary_key_list,
    DICTIONARY *prompt_dictionary );

/* Process */
LIST *dictionary_data_list(
    folder_attribute_primary_key_list
        /* attribute_name_list */,
    prompt_dictionary );

/* Usage */
boolean post_choose_isa_lookup_checkbox_checked(
    const char *WIDGET_LOOKUP_CHECKBOX_NAME,
    DICTIONARY *non_prefixed_dictionary );

/* Process */
boolean widget_checkbox_dictionary_checked(
    WIDGET_LOOKUP_CHECKBOX_NAME,
    non_prefixed_dictionary );

/* Usage */
DICTIONARY *post_choose_isa_query_dictionary(
    const char *QUERY_RELATION_OPERATOR_PREFIX,
    const char *QUERY_EQUAL,
    LIST *one_folder->
        folder_attribute_primary_key_list,
    LIST *post_choose_isa_primary_data_list );

/* Process */

/* Usage */
DICTIONARY *post_choose_isa_prompt_dictionary(
    LIST *folder->folder_attribute_primary_key_list,
    LIST *post_choose_isa_primary_data_list() );

/* Process */

/* Usage */
char *post_choose_isa_skip_system_string(
    const char *TABLE_INSERT_EXECUTABLE,
    const char *PROMPT_INSERT_EXECUTABLE,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *folder_appaserver_form,
    char *appaserver_error_filename );

/* Process */
int strcmp(
    FORM_APPASERVER_TABLE,
    folder_appaserver_form );

if ( strcmp() == 0 )
{
    char *execute_system_string_table_insert(
        TABLE_INSERT_EXECUTABLE,
        session_key,
        login_name,
        role_name,
        folder_name,
        FRAMESET_PROMPT_FRAME /* target_frame */,
        (char *)0 /* results_string */,
        (char *)0 /* error_string */,
        (char *)0 /* dictionary_separate_send_string */,
        appaserver_error_filespecification );
}
else
{
    char *execute_system_string_prompt_insert(
        PROMPT_INSERT_EXECUTABLE,
        session_key,
        login_name,
        role_name,
        folder_name,
        (char *)0 /* dictionary_separate_send_string */,
        appaserver_error_filespecification );
}

/* Driver */
if ( post_choose_isa->insert )
{
    char *insert_statement_error_string =
        insert_statement_execute(
            application_name,
            post_choose_isa->
                insert->
                insert_statement->
                insert_folder_statement_list,
            post_choose_isa->
                insert->
                appaserver_error_filespecification );

    char *execute_system_string_table_edit(
        TABLE_EDIT_EXECUTABLE,
        session_folder->session_key,
        session_folder->login_name,
        sessionn_folder->role_name,
        session_folder->folder_name,
        FRAMESET_PROMPT_FRAME /* target_frame */,
        post_choose_isa->insert->results_string,
        insert_statement_error_string,
        post_choose_isa->
            dictionary_separate_send_string,
        post_choose_isa->
            post_choose_isa_input->
            appaserver_error_filespecification );
}

if ( post_choose_isa->
        skip_system_string )
{
    system(
        post_choose_isa->
            skip_system_string );
}
else
if ( post_choose_isa->
        execute_system_string_table_edit )
{
    system(
        post_choose_isa->
            execute_system_string_table_edit );
}
else
{
    exit( 1 );
}
```

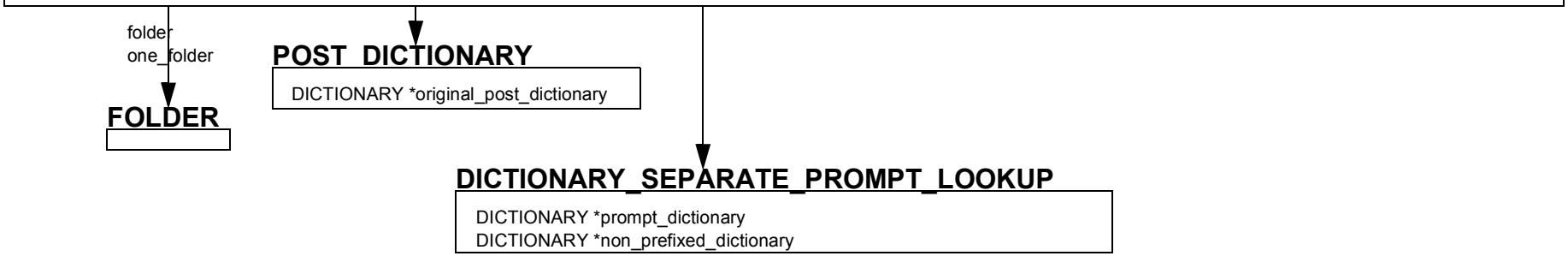
POST_CHOOSE_ISA_LOOKUP

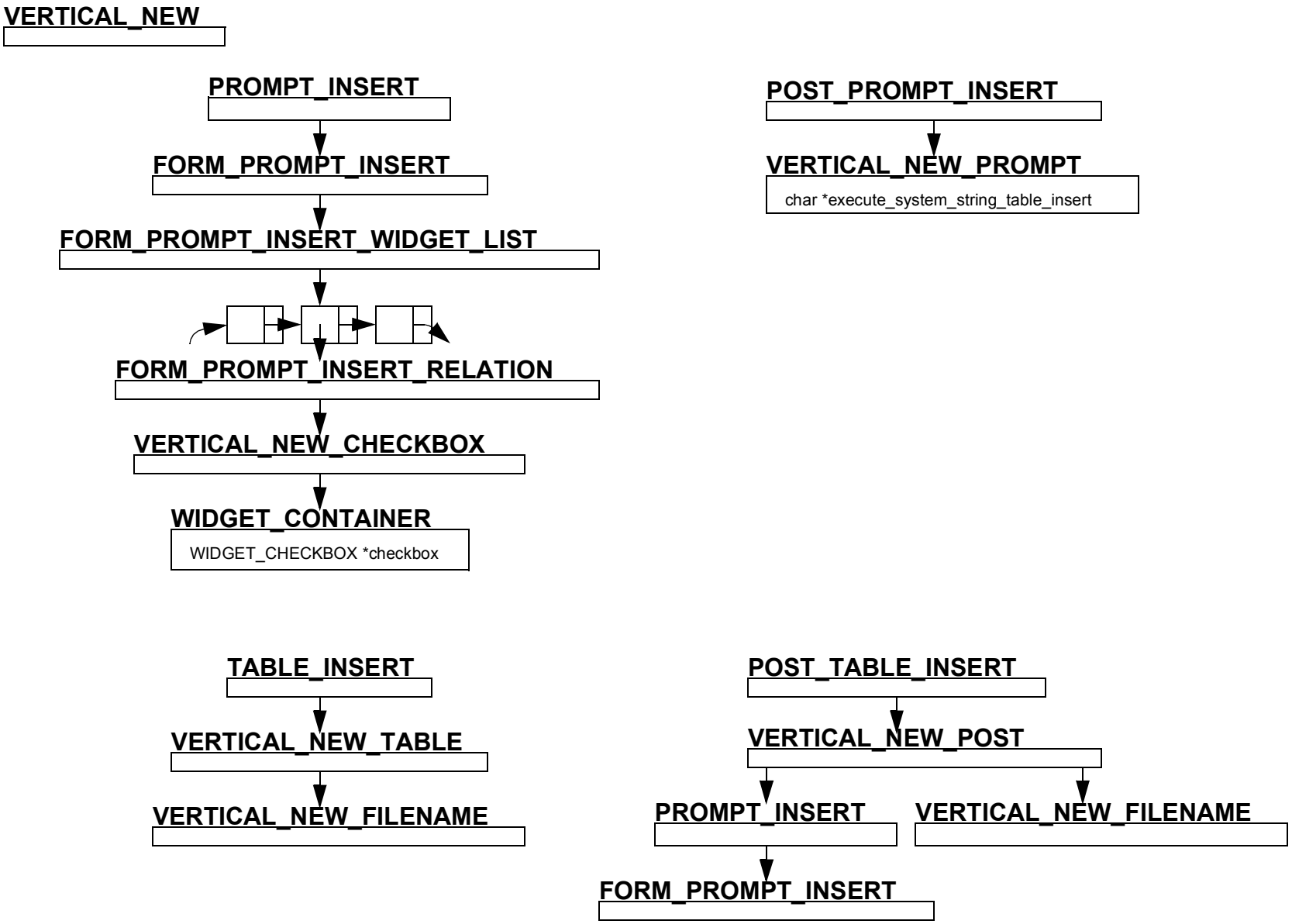
```
/* Usage */
POST_CHOOSE_ISA_LOOKUP *
    post_choose_isa_lookup_new(
        char *session_key,
        char *login_name,
        char *role_name,
        FOLDER *one_folder,
        DICTIONARY *prompt_dictionary,
        char *appaserver_error_filename,
        LIST *post_choose_isa_primary_data_list() );

/* Process */
POST_CHOOSE_ISA_LOOKUP *
    post_choose_isa_lookup_calloc(
        void );

DICTIONARY *post_choose_isa_query_dictionary(
    QUERY_RELATION_OPERATOR_PREFIX,
    QUERY_EQUAL,
    one_folder->
        folder_attribute_primary_key_list,
    post_choose_isa_primary_data_list );

DICTIONARY *dictionary_separate_send_dictionary(
    (DICTIONARY *)0 /* sort_dictionary */,
    DICTIONARY_SEPARATE_SORT_PREFIX,
    post_choose_isa_query_dictionary(),
    DICTIONARY_SEPARATE_QUERY_PREFIX,
    (DICTIONARY *)0 /* drillthru_dictionary */,
    DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
    (DICTIONARY *)0 /* ignore_dictionary */,
    DICTIONARY_SEPARATE_IGNORE_PREFIX,
    (DICTIONARY *)0 /* no_display_dictionary */,
    DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
    (DICTIONARY *)0 /* pair_dictionary */,
    DICTIONARY_SEPARATE_PAIR_PREFIX,
    (DICTIONARY *)0 /* non_prefixed_dictionary */ );

char *dictionary_separate_send_string(
    dictionary_separate_send_dictionary() );

char *execute_system_string_table_edit(
    TABLE_EDIT_EXECUTABLE,
    session_key,
    login_name,
    role_name,
    one_folder->folder_name,
    FRAMESET_PROMPT_FRAME /* target_frame */,
    (char *)0 /* results_string */,
    (char *)0 /* error_string */,
    dictionary_separate_send_string(),
    appaserver_error_filename );

/* Attributes */
DICTIONARY *post_choose_isa_query_dictionary;
DICTIONARY *dictionary_separate_send_dictionary;
char *dictionary_separate_send_string;
char *execute_system_string_table_edit;
```



POST_CHOOSE_ISA_INPUT

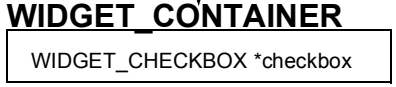
```
/* Usage */
POST_CHOOSE_ISA_INPUT *
   post_choose_isa_input_new(
      char *application_name,
      char *login_name,
      char *role_name,
      char *folder_name,
      char *one_folder_name );

/* Process */
POST_CHOOSE_ISA_INPUT *
   post_choose_isa_input_calloc(
      void );

/* Must preceed folder_name fetch due to caching */
FOLDER *one_folder =
   folder_fetch(
      one_folder_name,
      (LIST *)0 /* exclude_attribute_name_list */,
      /* Sets folder_attribute_primary_list */
      /* Sets folder_attribute_primary_key_list */
      /* Sets folder_attribute_name_list */
      1 /* fetch_folder_attribute_list */,
      1 /* fetch_attribute */,
      1 /* cache_boolean */ );

FOLDER *folder =
   folder_fetch(
      folder_name,
      (LIST *)0 /* exclude_attribute_name_list */,
      /* Sets folder_attribute_primary_list */
      /* Sets folder_attribute_primary_key_list */
      /* Sets folder_attribute_name_list */
      1 /* fetch_folder_attribute_list */,
      0 /* not fetch_attribute */
      1 /* cache_boolean */ );

POST_DICTIONARY *post_dictionary =
   post_dictionary_stdin_new(
      (char *)0 /* WIDGET_UPLOAD_RECALL_PREFIX */,
      (char *)0 /* application_name */,
      (char *)0 /* upload_directory */,
      (LIST *)0 /* upload_filename_list */ );

LIST *folder_attribute_date_name_list(
   one_folder->folder_attribute_list );

DICTIONARY_SEPARATE_PROMPT_LOOKUP *
   dictionary_separate =
      dictionary_separate_prompt_lookup_new(
         post_dictionary->original_post_dictionary,
         application_name,
         login_name,
         (LIST *)0 /* relation_mto1_list */,
         one_folder->folder_attribute_name_list,
         folder_attribute_date_name_list(),
         one_folder->folder_attribute_list );

char *appaserver_error_filespecification(
   application_name );

/* Attributes */
FOLDER *one_folder;
FOLDER *folder;
POST_DICTIONARY *post_dictionary;
LIST *folder_attribute_date_name_list;
DICTIONARY_SEPARATE_PROMPT_LOOKUP *
   dictionary_separate;
char *appaserver_error_filespecification;
```




VERTICAL_NEW
PROMPT_INSERT
FORM_PROMPT_INSERT
FORM_PROMPT_INSERT_WIDGET_LIST
FORM_PROMPT_INSERT_RELATION
VERTICAL_NEW_CHECKBOX
WIDGET_CONTAINER
WIDGET_CHECKBOX *checkbox
POST_PROMPT_INSERT
VERTICAL_NEW_PROMPT
char *execute_system_string_table_insert
TABLE_INSERT
VERTICAL_NEW_TABLE
VERTICAL_NEW_FILENAME
POST_TABLE_INSERT
VERTICAL_NEW_POST
PROMPT_INSERT
VERTICAL_NEW_FILENAME
FORM_PROMPT_INSERT

VERTICAL_NEW_CHECKBOX

```
/* Usage */
VERTICAL_NEW_CHECKBOX *
    vertical_new_checkbox_new(
        const char *VERTICAL_NEW_CHECKBOX_PREFIX,
        const char *VERTICAL_NEW_CHECKBOX_PROMPT,
        char *one_folder_name );

/* Process */
VERTICAL_NEW_CHECKBOX *
    vertical_new_checkbox_calloc(
        void );

char *vertical_new_checkbox_widget_name(
    const char *VERTICAL_NEW_CHECKBOX_PREFIX,
    char *one_folder_name );

WIDGET_CONTAINER *
    vertical_new_checkbox_widget_container(
        VERTICAL_NEW_CHECKBOX_PROMPT,
        vertical_new_checkbox_widget_name() );

/* Usage */
WIDGET_CONTAINER *
    vertical_new_checkbox_widget_container(
        const char *VERTICAL_NEW_CHECKBOX_PROMPT,
        char *vertical_new_checkbox_widget_name() );

/* Process */
WIDGET_CONTAINER *widget_container =
    widget_container_new(
        checkbox,
        vertical_new_checkbox_widget_name );

widget_container->checkbox->prompt =
    VERTICAL_NEW_CHECKBOX_PROMPT;

/* Attributes */
boolean role_folder_insert_boolean;
char *widget_name;
WIDGET_CONTAINER *widget_container;
```



VERTICAL_NEW_PROMPT

```
/* Usage */
VERTICAL_NEW_PROMPT *
   vertical_new_prompt_new(
      DICTIONARY *non_prefixed_dictionary,
      const char *
         VERTICAL_NEW_CHECKBOX_PREFIX,
      const char *
         VERTICAL_NEW_PROMPT_ONE_HIDDEN_LABEL,
      const char *
         VERTICAL_NEW_PROMPT_MANY_HIDDEN_LABEL,
      const char *TABLE_INSERT_EXECUTABLE,
      const char *FRAMESET_TABLE_FRAME,
      char *application_name,
      char *session_key,
      char *login_name,
      char *role_name,
      char *many_folder_name,
      DICTIONARY *drillthru_dictionary );

/* Process */
VERTICAL_NEW_PROMPT *vertical_new_prompt_calloc(
   void );

char *vertical_new_prompt_one_folder_name(
   VERTICAL_NEW_CHECKBOX_PREFIX,
   non_prefixed_dictionary );

if ( !vertical_new_prompt_one_folder_name() )
   return NULL;

DICTIONARY *vertical_new_prompt_dictionary(
   const char *
      VERTICAL_NEW_PROMPT_ONE_HIDDEN_LABEL,
   const char *
      VERTICAL_NEW_PROMPT_MANY_HIDDEN_LABEL,
   char *many_folder_name,
   char *vertical_new_prompt_one_folder_name() );

char *dictionary_separate_send_string(
   dictionary_separate_send_dictionary(
      ...,
      drillthru_dictionary,
      ...,
      vertical_new_prompt_dictionary()
         /* non_prefixed_dictionary */ ) );

char *execute_system_string_table_insert(
   TABLE_INSERT_EXECUTABLE,
   session_key,
   login_name,
   role_name,
   one_folder_name /* folder_name */,
   FRAMESET_EDIT_FRAME /* target_frame */,
   (char *)0 /* results_string */,
   (char *)0 /* error_string */,
   dictionary_separate_send_string(),
      appaserver_error_filename(
         application_name ) );

/* Usage */
char *vertical_new_prompt_one_folder_name(
   const char *VERTICAL_NEW_CHECKBOX_PREFIX,
   DICTIONARY *non_prefixed_dictionary );

/* Process */
LIST *dictionary_prefix_key_list(
   VERTICAL_NEW_CHECKBOX_PREFIX,
   non_prefixed_dictionary );

char *list_first( dictionary_prefix_key_list() );

if ( list_first() )
{
   char *one_folder_name =
      list_first() +
      strlen( VERTICAL_NEW_CHECKBOX_PREFIX );
}

/* Attributes */
char *one_folder_name;
char *many_folder_name;
char *execute_system_string_table_insert;
```

VERTICAL_NEW_TABLE (1)

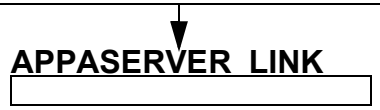
```
/* Usage */
VERTICAL_NEW_TABLE *vertical_new_table_new(
    const char *
        VERTICAL_NEW_PROMPT_ONE_HIDDEN_LABEL,
    const char *
        VERTICAL_NEW_PROMPT_MANY_HIDDEN_LABEL,
    const char *FRAMESET_PROMPT_FRAME,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *data_directory,
    DICTIONARY *non_prefixed_dictionary,
    boolean menu_horizontal_boolean() );

/* Process */
VERTICAL_NEW_TABLE *vertical_new_table_calloc(
    void );

char *vertical_new_table_one_folder_name(
    VERTICAL_NEW_PROMPT_ONE_HIDDEN_LABEL,
    non_prefixed_dictionary );

if ( !vertical_new_table_one_folder_name() )
        return NULL;

char *vertical_new_table_many_folder_name(
    VERTICAL_NEW_PROMPT_MANY_HIDDEN_LABEL,
    non_prefixed_dictionary );

if ( !vertical_new_table_many_folder_name() )
        exit( 1 );

VERTICAL_NEW_FILENAME *vertical_new_filename =
    vertical_new_filename_new(
            application_name,
            session_key,
            data_directory,
            "vertical_table"
                /* filename_stem */ );

char *vertical_new_table_onload_javascript_string(
    FRAMESET_PROMPT_FRAME,
    vertical_new_blank_filename->prompt_filename );

if ( menu_horizontal_boolean )
{
    FOLDER_MENU *folder_menu =
        folder_menu_fetch(
            application_name,
            session_key,
            role_name,
            data_directory,
            0 /* not folder_menu_remove_boolean */ );

    menu =
        menu_new(
            application_name,
            session_key,
            login_name,
            role_name,
            1 /* application_menu_horizontal_boolean */,
            folder_menu->count_list );
}

DOCUMENT *document =
    document_new(
        application_name,
        application_title_string( application_name ),
        (char *)0 /* title_string */,
        (char *)0 /* sub_title_string */,
        (char *)0 /* sub_sub_title_string */,
        (char *)0 /* notepad */,
        (char *)0 /* onload_javascript_string */,
        (menu) ? menu->html : (char *)0,
        document_head_menu_setup_string(
            (menu) ? 1 : 0 /* menu_boolean */ ),
        (char *)0 /* calendar_setup_string */,
        javascript_include_string(
            application_name,
            (LIST *)0 /* javascript_filename_list */ ) );

/* Attributes */
char *many_folder_name;
char *one_folder_name;
VERTICAL_NEW_FILENAME *vertical_new_filename;
char *onload_javascript_string;
FOLDER_MENU *folder_menu;
MENU *menu;
DOCUMENT *document;
```

VERTICAL_NEW_TABLE (2)

```
/* Usage */
char *vertical_new_table_one_folder_name(
    const char *
        VERTICAL_NEW_PROMPT_ONE_HIDDEN_LABEL,
    DICTIONARY *non_prefixed_dictionary );

/* Process */
char *dictionary_get(
    VERTICAL_NEW_PROMPT_ONE_HIDDEN_LABEL,
    non_prefixed_dictionary );

/* Usage */
char *vertical_new_table_many_folder_name(
    const char *
        VERTICAL_NEW_PROMPT_MANY_HIDDEN_LABEL,
    DICTIONARY *non_prefixed_dictionary );

/* Process */
char *dictionary_get(
    VERTICAL_NEW_PROMPT_MANY_HIDDEN_LABEL,
    non_prefixed_dictionary );

/* Usage */
char *vertical_new_table_onload_javascript_string(
    const char *FRAMESET_PROMPT_FRAME,
    char *vertical_new_filename->
        prompt_filename );

/* Process */
sprintf(
    static char javascript_string[],
    "window.open( '%s', '%s' )",
    prompt_filename,
    FRAMESET_PROMPT_FRAME );

/* Usage */
boolean vertical_new_table_participating_boolean(
    const char *
        VERTICAL_NEW_PROMPT_ONE_HIDDEN_LABEL,
    DICTIONARY *non_prefixed_dictionary );

/* Process */
boolean dictionary_key_exists(
    non_prefixed_dictionary,
    VERTICAL_NEW_PROMPT_ONE_HIDDEN_LABEL
        /* key */ );

/* Driver */
void vertical_new_table_blank_prompt_frame(
    char *vertical_new_table->
        vertical_new_filename->
        output_filename,
    DOCUMENT *vertical_new_table->
        document );

/* Process */
FILE *appaserver_output_file( output_filename );

fprintf(
    appaserver_output_file(),
    "%s\n%s\n%s\n%s\n%s\n%s\n",
    document->html,
    document->document_head->html,
    document_head_close_tag(),
    document->document_body->html,
    document_body_close_tag(),
    document_close_tag() );

fclose( appaserver_output_file() );
```

VERTICAL_NEW_FILENAME

```
VERTICAL_NEW_FILENAME *
   vertical_new_filename_new(
      char *application_name,
      char *session_key,
      char *data_directory,
      const char *filename_stem );

/* Process */
VERTICAL_NEW_FILENAME *
   vertical_new_filename_calloc(
      void );

APPASERVER_LINK *appaserver_link =
   appaserver_link_new(
      application_http_prefix(
         application_ssl_support_boolean(
            application_name ) ),
      application_server_address(),
      data_directory,
      filename_stem,
      application_name,
      0 /* process_id */,
      session_key,
      (char *)0 /* begin_date_string */,
      (char *)0 /* end_date_string */,
      "html" /* extension */ );

char *output_filename =
   appaserver_link->
      appaserver_link_output->
      filename;

char *prompt_filename =
   appaserver_link->
      appaserver_link_prompt->
      filename;

/* Attributes */
APPASERVER_LINK *appaserver_link;
char *output_filename;
char *prompt_filename;
```



VERTICAL_NEW_POST

```
/* Usage */
VERTICAL_NEW_POST *vertical_new_post_new(
    const char *
        VERTICAL_NEW_PROMPT_MANY_HIDDEN_LABEL,
    const char *FRAMESET_PROMPT_FRAME,
    const char *
        DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    DICTIONARY *drillthru_dictionary,
    DICTIONARY *non_prefixed_dictionary,
    char *appaserver_parameter->data_directory );

/* Process */
VERTICAL_NEW_POST *vertical_new_post_calloc(
    void );

char *vertical_new_post_many_folder_name(
    const char *
        VERTICAL_NEW_PROMPT_MANY_HIDDEN_LABEL,
    DICTIONARY *non_prefixed_dictionary );

if ( !vertical_new_post_many_folder_name() )
        return NULL;

boolean application_menu_horizontal_boolean(
    application_name );

DICTIONARY *original_post_dictionary =
    vertical_new_post_original_dictionary(
        DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
        drillthru_dictionary,
        non_prefixed_dictionary );

VERTICAL_NEW_FILENAME *vertical_new_filename =
    vertical_new_filename_new(
        application_name,
        session_key,
        data_directory,
        "vertical_post"
            /* filename_stem */ );

char *vertical_new_post_onload_javascript_string(
    FRAMESET_PROMPT_FRAME,
    vertical_new_filename->prompt_filename );

/* Usage */
char *vertical_new_post_onload_javascript_string(
    const char *FRAMESET_PROMPT_FRAME,
    char *prompt_filename );

/* Process */
sprintf(
    static char javascript_string[],
    "window.open( '%s', '%s' )",
    prompt_filename,
    FRAMESET_PROMPT_FRAME );

/* Usage */
DICTIONARY *vertical_new_post_original_dictionary(
    const char *
        DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
    DICTIONARY *drillthru_dictionary,
    DICTIONARY *non_prefixed_dictionary );

/* Process */
DICTIONARY *original_post_dictionary =
    dictionary_medium();

(void)dictionary_append(
    original_post_dictionary,
    non_prefixed_dictionary );

if ( dictionary_length( drillthru_dictionary ) )
{
    DICTIONARY *dictionary_prefix(
        drillthru_dictionary,
        DICTIONARY_SEPARATE_DRILLTHRU_PREFIX );
    (void)dictionary_set_dictionary(
        original_post_dictionary,
        dictionary_prefix() );
}

/* Driver */
void vertical_new_post_prompt_insert_create(
    VERTICAL_NEW_FILENAME *
        vertical_new_post->
            vertical_new_filename,
    PROMPT_INSERT *
        vertical_new_post->
            prompt_insert );

/* Process */
FILE *appaserver_output_file(
    vertical_new_post->
        vertical_new_filename->
        output_filename );

fprintf(
    appaserver_output_file(),
    "%s\n",
    vertical_new_post->
        prompt_insert->
        document_form_html );

fclose( appaserver_output_file() );

/* Attributes */
char *many_folder_name;
char *data_directory;
boolean application_menu_horizontal_boolean;
DICTIONARY *original_post_dictionary;
VERTICAL_NEW_FILENAME *vertical_new_filename;
char *onload_javascript_string;

/* Set externally */
PROMPT_INSERT *prompt_insert;
```

ROW_SECURITY (1)

```
/* Usage */
ROW_SECURITY *row_security_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    LIST *folder_attribute_non_primary_name_list,
    LIST *relation_mto1_list,
    LIST *relation_mto1_isa_list,
    LIST *relation_one2m_join_list,
    LIST *folder_attribute_append_isa_list,
    char *folder->post_change_javascript,
    boolean table_edit_viewonly_boolean,
    DICTIONARY *drop_down_dictionary,
    LIST *no_display_name_list,
    LIST *query_select_name_list,
    LIST *viewonly_attribute_name_list,
    char *table_edit_state,
    /* Null if not participating */
    ROW_SECURITY_ROLE_UPDATE_LIST *
        row_security_role_update_list,
    int query_row_list_length );

/* Process */
ROW_SECURITY *row_security_calloc(
    void );

boolean row_security_regular_boolean(
    const char *APPASERVER_UPDATE_STATE,
    char *table_edit_state );

boolean row_security_viewonly_relation_boolean(
    boolean table_edit_viewonly_boolean,
    ROW_SECURITY_ROLE_UPDATE_LIST *
        row_security_role_update_list,
    boolean row_security_regular_boolean() );

boolean row_security_viewonly_no_relation_boolean(
    boolean row_security_regular_boolean() );

if ( row_security_regular_boolean() )
{
    APPLICATION *application =
        application_fetch(
            application_name );

    ROW_SECURITY_REGULAR_WIDGET_LIST *
        regular_widget_list =
            row_security_regular_widget_list_new(
                application_name,
                session_key,
                login_name,
                role_name,
                folder_name,
                folder_attribute_non_primary_name_list,
                relation_mto1_list,
                relation_mto1_isa_list,
                relation_one2m_join_list,
                folder_attribute_append_isa_list,
                post_change_javascript,
                drop_down_dictionary,
                no_display_name_list,
                query_select_name_list,
                viewonly_attribute_name_list,
                row_security_role_update_list,
                query_row_list_length,
                application->
                    max_query_rows_for_drop_downs,
                application->
                    max_drop_down_size );
}

if ( row_security_viewonly_relation_boolean() )
{
    ROW_SECURITY_VIEWONLY_WIDGET_LIST *
        viewonly_widget_list =
            row_security_viewonly_widget_list_new(
                application_name,
                session_key,
                login_name,
                relation_mto1_list,
                relation_mto1_isa_list,
                relation_one2m_join_list,
                folder_attribute_append_isa_list,
                no_display_name_list,
                query_select_name_list,
                viewonly_attribute_name_list,
                row_security_role_update_list );
}
else
if ( row_security_viewonly_no_relation_boolean() )
{
    ROW_SECURITY_VIEWONLY_WIDGET_LIST *
        viewonly_widget_list =
            row_security_viewonly_widget_list_new(
                application_name,
                session_key,
                login_name,
                (LIST *)0 /* relation_mto1_list */,
                (LIST *)0 /* relation_mto1_isa_list */,
                relation_one2m_join_list,
                folder_attribute_append_isa_list,
                no_display_name_list,
                query_select_name_list,
                viewonly_attribute_name_list,
                row_security_role_update_list );
}

/* Attributes */
boolean regular_boolean;
boolean viewonly_relation_boolean;
boolean viewonly_no_relation_boolean;
APPLICATION *application;
ROW_SECURITY_REGULAR_WIDGET_LIST *
    regular_widget_list;
ROW_SECURITY_VIEWONLY_WIDGET_LIST *
    viewonly_widget_list;
```

APPLICATION **ROW_SECURITY_REGULAR_WIDGET_LIST** **ROW_SECURITY_VIEWONLY_WIDGET_LIST**

ROW_SECURITY (2)

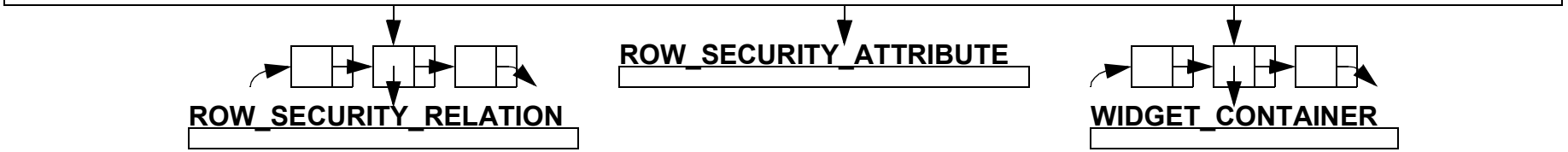
```
/* Usage */
LIST *row_security_join_container_list(
    LIST *relation_one2m_join_list );

/* Process */
for RELATION_ONE2M *relation_one2m in
        relation_one2m_join_list
{
    list_set(
        LIST *join_widget_container_list,
            widget_container_new(
                table_data, (char *)0 ) );

    list_set(
        LIST *join_widget_container_list,
        ( WIDGET_CONTAINER *
            drop_down_widget_container =
                widget_container_new(
                    drop_down,
                relation_one2m->many_folder_name ) ) );

    drop_down_widget_container->
        drop_down->
        display_size = 3;

    drop_down_widget_container->
        drop_down->
        multi_select_boolean = 1;

    LIST *folder_attribute_non_primary_name_list(
        relation_one2m->
            many_folder->
            folder_attribute_list );

    drop_down_widget_container->
        drop_down->
        foreign_key_list =
            folder_attribute_non_primary_name_list();

    drop_down_widget_container->
        heading_string =
            relation_one2m->many_folder_name;
}

/* Usage */
LIST *row_security_role_update_container_list(
    char *attribute_not_null );

/* Process */
LIST *list = list_new();

list_set(
    list,
    widget_container_new(
        table_data, (char *)0 ) );

WIDGET_CONTAINER *widget_container =
    widget_container_new(
        character,
        attribute_not_null );

widget_container->heading_string = attribute_not_null;
widget_container->text->viewonly = 1;

list_set( list, widget_container );

/* Usage */
LIST *row_security_ajax_client_list(
    ROW_SECURITY *row_security );

/* Process */
if ( row_security->regular_widget_list )
{
    LIST *row_security_relation_ajax_client_list(
        row_security->
            regular_widget_list->
            row_security_relation_list );
}
```

ROW_SECURITY_REGULAR_WIDGET_LIST (1)

```
/* Usage */
ROW_SECURITY_REGULAR_WIDGET_LIST *
    row_security_regular_widget_list_new(
        char *application_name,
        char *session_key,
        char *login_name,
        char *role_name,
        char *folder_name,
        LIST *folder_attribute_non_primary_name_list,
        LIST *relation_mto1_list,
        LIST *relation_mto1_isa_list,
        LIST *relation_one2m_join_list,
        LIST *folder_attribute_append_isa_list,
        char *post_change_javascript,
        DICTIONARY *drop_down_dictionary,
        LIST *no_display_name_list,
        LIST *query_select_name_list,
        LIST *viewonly_attribute_name_list,
        /* Null if not participating */
        ROW_SECURITY_ROLE_UPDATE_LIST *
            row_security_role_update_list,
        int query_row_list_length,
        int max_query_rows_for_drop_downs,
        int max_drop_down_size );

/* Process */
ROW_SECURITY_REGULAR_WIDGET_LIST *
    row_security_regular_widget_list_calloc(
        void );

LIST *widget_container_list = list_new();

for FOLDER_ATTRIBUTE *folder_attribute in
    folder_attribute_append_isa_list
{
    if ( !list_string_exists(
            folder_attribute->attribute_name,
            query_select_name_list ) )
    {
            continue;
    }

    if ( list_string_exists(
        folder_attribute->attribute_name,
        no_display_name_list ) )
    {
        if ( folder_attribute->primary_key_index )
        {
            list_set(
                widget_container_list,
                widget_container_new(
                    hidden,
                    folder_attribute->attribute_name ) );
        }
        continue;
    }

    boolean list_string_exists(
        vewionly_attribute_name_list,
        folder_attribute->attribute_name );

    if (    !list_string_exists()
    &&   ( ROW_SECURITY_RELATION *
                row_security_relation =
                    row_security_relation_new(
                        application_name,
                        session_key,
                        login_name,
                        role_name,
                        folder_name
                            /* many_folder_name */,
                        folder_attribute_non_primary_name_list,
                        relation_mto1_list
                            /* many_folder_relation_mto1_list */,
                        relation_mto1_isa_list,
                        folder_attribute->attribute_name,
                        folder_attribute->primary_key_index,
                        post_change_javascript,
                        drop_down_dictionary,
                        security_entity,
                        0 /* not viewonly_boolean */,
                        query_row_list_length,
                        max_query_rows_for_drop_downs,
                        max_drop_down_size,
                        row_security_relation_list ) ) )
    {
        list_set(
            row_security_relation_list,
            row_security_relation );

        list_set(
            widget_container_list,
                widget_container_new(
                    table_data,
                        (char *)0 /* widget_name */ ) );

        list_set(
            widget_container_list,
            row_security_relation->
                widget_container );

        if (    row_security_relation->viewonly_boolean
        &&   folder_attribute->primary_key_index )
        {
            list_set(
                widget_container_list,
                widget_container_new(
                    hidden,
                    row_security_relation->
                        relation_mto1->
                        relation_name
                        /* widget_name */ ) );
        }

        continue;
    }

    if ( ( ROW_SECURITY_ATTRIBUTE *
            row_security_attribute =
                row_security_attribute_new(
                    application_name,
                    session_key,
                    login_name,
                    viewonly_attribute_name_list,
                    folder_attribute->attribute_name,
                    folder_attribute->primary_key_index,
                    folder_attribute->prompt,
                    folder_attribute->
                        attribute->
                        datatype_name,
                    folder_attribute->
                        attribute->
                        width,
                    post_change_javascript,
                    row_security_relation_list ) ) )
    {
        list_set_list(
            widget_container_list,
            row_security_attribute->
                widget_container_list );
    }
}
```

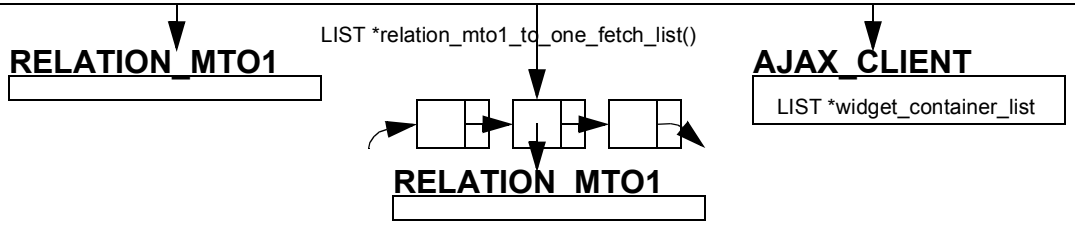
ROW_SECURITY_REGULAR_WIDGET_LIST (2)

```
/* Process (continued) */
if (    row_security_role_update_list
&&   row_security_role_update_list->
           row_security_role_update_relation )
{
    list_set_list(
        widget_container_list,
        LIST *row_security_role_update_container_list(
            row_security_role_update_list->
                attribute_not_null ) );
}

if ( list_length( relation_one2m_join_list ) )
{
    list_set_list(
        widget_container_list,
        LIST *row_security_join_container_list(
            relation_one2m_join_list ) );
}

/* Attributes */
LIST *row_security_relation_list;
ROW_SECURITY_ATTRIBUTE *row_security_attribute;
LIST *widget_container_list;
```



ROW_SECURITY_RELATION (1)

```
/* Usage */
ROW_SECURITY_RELATION *
   row_security_relation_new(
      char *application_name,
      char *session_key,
      char *login_name,
      char *role_name,
      char *many_folder_name,
      LIST *folder_attribute_non_primary_name_list,
      LIST *many_folder_relation_mto1_list,
      LIST *relation_mto1_isa_list,
      char *attribute_name,
      int primary_key_index,
      char *post_change_javascript,
      DICTIONARY *drop_down_dictionary,
      boolean viewonly_boolean,
      int query_row_list_length,
      int max_query_rows_for_drop_downs,
      int max_drop_down_size,
      LIST *row_security_relation_list );

/* Process */
if (    !list_length( many_folder_relation_mto1_list )
&&   !list_length( relation_mto1_isa_list ) )
{
   return NULL;
}

if ( row_security_relation_attribute_name_exists(
      char *attribute_name,
      LIST *row_security_relation_list ) )
{
   return NULL;
}

ROW_SECURITY_RELATION *
   row_security_relation_calloc(
      void );

RELATION_MTO1 *relation_mto1 =
   relation_mto1_consumes(
      attribute_name,
      many_folder_relation_mto1_list );

if ( !relation_mto1 )
{
   relation_mto1 =
      relation_mto1_isa_consumes(
         attribute_name,
         relation_mto1_isa_list );

   if ( !relation_mto1 ) return NULL;
}

if (    primary_key_index
&&   relation_mto1->
         relation->
         relation_type_isa )
{
   return NULL;
}

if ( viewonly_boolean )
{
   WIDGET_CONTAINER *
      widget_container =
         widget_container_new(
            non_edit_text,
            relation_mto1->relation_name );

   goto row_security_relation_heading_set;
}

LIST *relation_mto1_to_one_fetch_list(
   relation_mto1->one_folder_name,
   relation_mto1->
      one_folder->
      folder_attribute_primary_key_list );

if ( relation_mto1->
      relation->
      ajax_fill_drop_down )
{
   AJAX_CLIENT *ajax_client =
      /* Returns null if not participating */
      ajax_client_relation_mto1_new(
         relation_mto1,
         relation_mto1_to_one_fetch_list(),
      0 /* not top_select_boolean */ );
}

if ( ajax_client )
{
   char *relation_mnemonic(
      relation_mto1->relation_foreign_key_list );

   WIDGET_CONTAINER *widget_container =
      row_security_relation_ajax_widget_container(
         relation_mto1->
            relation_mnemonic()
               /* widget_name */,
         ajax_client->
            widget_container_list );

   goto row_security_relation_heading_set;
}
```

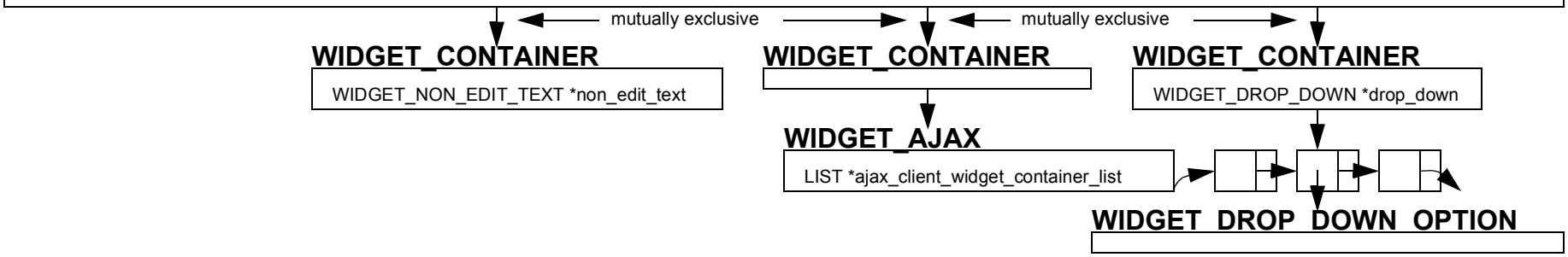


ROW_SECURITY_RELATION (2)

```
/* Process (continued) */
LIST *relation_mto1_common_name_list(
    folder_attribute_non_primary_name_list,
    relation_mto1->
        relation->
        copy_common_columns,
    relation_mto1->
        one_folder->
        folder_attribute_name_list );

QUERY_DROP_DOWN *query_drop_down_new(
    application_name,
    session_key,
    login_name,
    role_name,
    APPASERVER_UPDATE_STATE,
    many_folder_name,
    relation_mto1->one_folder_name,
    relation_mto1->relation->related_attribute_name
        /* related_column */,
    relation_mto1->
        one_folder->
        folder_attribute_list,
    relation_mto1->
        one_folder->
        populate_drop_down_process_name,
    relation_mto1_to_one_fetch_list(),
    drop_down_dictionary /* dictionary */,
    (SECURITY_ENTITY *)0 /* security_entity */,
    relation_mto1_common_name_list() );

int row_security_relation_viewonly_boolean(
    TABLE_EDIT_FORCE_DROP_DOWN_ROW_COUNT,
    int query_row_list_length,
    int max_query_rows_for_drop_downs,
    int max_drop_down_size,
    int list_length(
        query_drop_down_new()->delimited_list )
        /* delimited_list_length */ );

if ( row_security_relation_viewonly_boolean() )
{
    WIDGET_CONTAINER *
        widget_container =
            widget_container_new(
                non_edit_text,
                relation_mto1->relation_name );
}
else
{
    WIDGET_CONTAINER *
        widget_container =
        row_security_relation_drop_down_widget_new(
            application_name,
            login_name,
            post_change_javascript,
            relation_mto1,
            query_drop_down->
                delimited_list );
}

row_security_relation_heading_set:

widget_container->heading_string =
    relation_mto1->relation_prompt;

/* Usage */
WIDGET_CONTAINER *
    row_security_relation_drop_down_widget_new(
        char *application_name,
        char *login_name,
        char *post_change_javascript,
        RELATION_MTO1 *relation_mto1,
        LIST *query_drop_down->delimited_list );

/* Process */
WIDGET_CONTAINER *
    widget_container =
        widget_container_new(
            drop_down,
            relation_mto1->relation_name
                /* widget_name */ );

widget_container->
    drop_down->
    null_boolean = 1;

widget_container->
    drop_down->
    top_select_boolean = 1;

widget_container->
    drop_down->
    display_size = 1;

widget_container->
    drop_down->
    post_change_javascript =
        post_change_javascript;

widget_container->
    drop_down->
    foreign_key_list =
        relation_mto1->relation_foreign_key_list;

widget_container->
    drop_down->
    widget_drop_down_option_list =
        LIST *widget_drop_down_option_list(
            ATTRIBUTE_MULTI_KEY_DELIMITER,
            WIDGET_DROP_DOWN_LABEL_DELIMITER,
            WIDGET_DROP_DOWN_EXTRA_DELIMITER,
            WIDGET_DROP_DOWN_DASH_DELIMITER,
            delimited_list,
            relation_mto1->
                one_folder->
                no_initial_capital );
```

ROW_SECURITY_RELATION (3)

```
/* Usage */
WIDGET_CONTAINER *
   row_security_relation_ajax_widget_container(
      char *widget_name,
      LIST *ajax_client_widget_container_list );

/* Process */
WIDGET_CONTAINER *widget_container =
   widget_container_ajax_new(
      widget_name,
      ajax_client_widget_container_list );

/* Usage */
LIST *row_security_relation_ajax_client_list(
   LIST *row_security_relation_list );

/* Process */
LIST *ajax_client_list = list_new();

for ROW_SECURITY_RELATION *row_security_relation in
   row_security_relation_list
{
   if ( row_security_relation->ajax_client )
   {
      list_set(
         ajax_client_list,
         row_security_relation->
            ajax_client );
   }

   /* Attributes */
   RELATION_MTO1 *relation_mto1;
   LIST *relation_mto1_to_one_fetch_list;
   LIST *relation_mto1_common_name_list;
   QUERY_DROP_DOWN *query_drop_down;
   boolean viewonly_boolean;
   AJAX_CLIENT *ajax_client;
   WIDGET_CONTAINER *widget_container;
```



ROW_SECURITY_ATTRIBUTE (1)

```
/* Usage */
ROW_SECURITY_ATTRIBUTE *
    row_security_attribute_new(
        char *application_name,
        char *session_key,
        char *login_name,
        LIST *viewonly_attribute_name_list,
        char *attribute_name,
        int primary_key_index,
        char *folder_attribute_prompt,
        char *datatype_name,
        int attribute_width,
        char *post_change_javascript,
        LIST *row_security_relation_list );

/* Process */
if ( row_security_relation_attribute_name_exists(
    attribute_name,
    row_security_relation_list ) )
{
    return NULL;
}

ROW_SECURITY_ATTRIBUTE *
    row_security_attribute_calloc(
        void );

LIST *widget_container_list = list_new();

list_set(
    widget_container_list,
    ( WIDGET_CONTAINER *widget_container =
        widget_container_new(
            table_data,
            (char *)0 /* widget_name */ ) ) );

boolean attribute_is_number( datatype_name );

if ( attribute_is_number() )
{
    widget_container->table_data->align_right = 1;
}

boolean row_security_attribute_non_edit_boolean(
    viewonly_attribute_name_list,
    attribute_name,
    datatype_name );

if ( attribute_is_password( datatype_name ) )
{
    ROW_SECURITY_PASSWORD *
        row_security_password_new(
            attribute_name,
            attribute_width,
            folder_attribute_prompt,
            row_security_attribute_non_edit_boolean()
                /* viewonly_boolean */ );

    list_set_list(
        widget_container_list,
        row_security_password_new()->
            widget_container_list );

    return this;
}

if ( row_security_attribute_non_edit_boolean() )
{
    widget_container =
        widget_container_new(
          non_edit_text,
          attribute_name );

    widget_container->
        non_edit_text->
        no_initial_capital_boolean = 1;

    if ( primary_key_index )
    {
        list_set(
            widget_container_list,
            widget_container_new(
                hidden,
                attribute_name ) );
    }
}
else
if ( attribute_is_yes_no( attribute_name ) )
{
    widget_container =
        widget_container_new(
            yes_no,
            attribute_name );

    widget_container->
        yes_no->
        post_change_javascript =
            post_change_javascript;
}
else
if ( attribute_is_notepad(
        ATTRIBUTE_DATATYPE_NOTEPAD,
        datatype_name ) )
{
    widget_container =
        widget_container_new(
            notepad,
            attribute_name );

    widget_container->
        notepad->
        post_change_javascript =
            post_change_javascript;

    widget_container->
        notepad->
        attribute_size = attribute_width;

    widget_container->
        notepad->
        null_to_slash = 1;
}
else
```

ROW_SECURITY_ATTRIBUTE (2)

```
/* Process (continued) */
if ( attribute_is_encrypt( datatype_name ) )
{
    widget_container =
        widget_container_new(
            encrypt,
            attribute_name );

    widget_container->
        encrypt->
        attribute_width_max_length = attribute_width;
}
else
if ( attribute_is_upload( datatype_name ) )
{
    widget_container =
        widget_container_new(
            upload,
            attribute_name );

    widget_container->
        upload->
        application_name = application_name;

    widget_container->
        upload->
        session_key = session_key;
}
else
if (    attribute_is_date( datatype_name )
||      attribute_is_date_time( datatype_name )
||      attribute_is_current_date_time( datatype_name ) )
{
    widget_container =
        widget_container_new(
            widget_date,
            attribute_name );

    widget_container->
        date->
        datatype_name = datatype_name;

    widget_container->
        date->
        application_name = application_name;

    widget_container->
        date->
        login_name = login_name;

    widget_container->
        date->
        post_change_javascript =
            post_change_javascript;

    widget_container->
        date->
        attribute_width_max_length = attribute_width;

    int widget_text_display_size(
        WIDGET_TEXT_DEFAULT_MAX_LENGTH,
        attribute_width );

    widget_container->
        date->
        display_size =
            widget_text_display_size();

    widget_container->date->null_to_slash_boolean = 1;
}
else
{
    widget_container =
        widget_container_new(
            character,
            attribute_name );

    widget_container->
        text->
        post_change_javascript =
            post_change_javascript;

    widget_container->
        text->
        datatype_name =
            datatype_name;

    widget_container->
        text->
        attribute_width_max_length =
            attribute_width;

    int widget_text_display_size(
        WIDGET_TEXT_DEFAULT_MAX_LENGTH,
        attribute_width );

    widget_container->
        text->
        widget_text_display_size =
            widget_text_display_size();

    widget_container->text->null_to_slash = 1;

    widget_container->text->prevent_carrot_boolean =
        (boolean)primary_key_index;
}

widget_container->heading_string =
    folder_attribute_prompt;

list_set(
    widget_container_list,
    widget_container );
```

ROW_SECURITY_ATTRIBUTE (3)

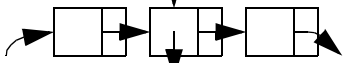
```
/* Usage */
boolean row_security_attribute_non_edit_boolean(
    LIST *viewonly_attribute_name_list,
    char *attribute_name,
    char *datatype_name );

/* Process */
boolean list_string_exists(
    attribute_name,
    viewonly_attribute_name_list );

if ( list_string_exists() ) return 1;

boolean attribute_is_timestamp(
    datatype_name );

if ( attribute_is_timestamp() ) return 1;

return 0;

/* Attributes */
LIST *widget_container_list;
boolean non_edit_boolean;
ROW_SECURITY_PASSWORD *row_security_password;
WIDGET_CONTAINER *widget_container;
```

ROW_SECURITY_PASSWORD

WIDGET_CONTAINER

mutually exclusive

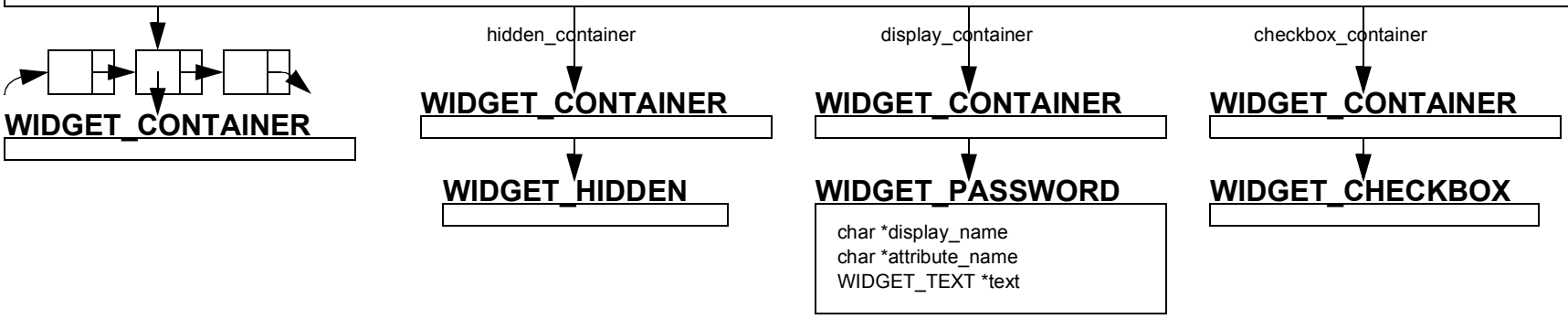
- WIDGET_NON_EDIT_TEXT *non_edit_text
- WIDGET_YES_NO *yes_no
- WIDGET_TEXT *text
- WIDGET_NOTEPAD *notepad
- WIDGET_UPLOAD *upload
- WIDGET_DATE *date



WIDGET_CONTAINER

ROW_SECURITY_PASSWORD (1)

```
/* Usage */
ROW_SECURITY_PASSWORD *
   row_security_password_new(
      char *attribute_name,
      int attribute_width,
      char *folder_attribute_prompt,
      boolean viewonly_boolean );

/* Process */
ROW_SECURITY_PASSWORD *
   row_security_password_calloc(
      void );

LIST *widget_container_list = list_new();

WIDGET_CONTAINER *widget_container_new(
   hidden,
   attribute_name );

list_set(
   widget_container_list,
      widget_container_new() );

char *widget_password_display_name(
   attribute_name );

WIDGET_CONTAINER *
   row_security_password_display_container(
      attribute_name,
      attribute_width,
      folder_attribute_prompt,
      viewonly_boolean,
      widget_password_display_name() );

list_set(
   widget_container_list,
   row_security_password_display_container() );

WIIDGET_CONTAINER *widget_container_new(
   line_break,
   (char *)0 /* widget_name */ );

list_set(
   widget_container_list,
   widget_container_new() );

WIDGET_CONTAINER *
   row_security_password_checkbox_container(
      attribute_name,
      widget_password_display_name() );

list_set(
   widget_container_list,
   widget_password_checkbox_container() );

/* Attributes */
char *attribute_name;
LIST *widget_container_list;
WIDGET_CONTAINER *hidden_container;
char *widget_password_display_name;
WIDGET_CONTAINER *display_container;
WIDGET_CONTAINER *checkbox_container;
```



ROW_SECURITY_PASSWORD (2)

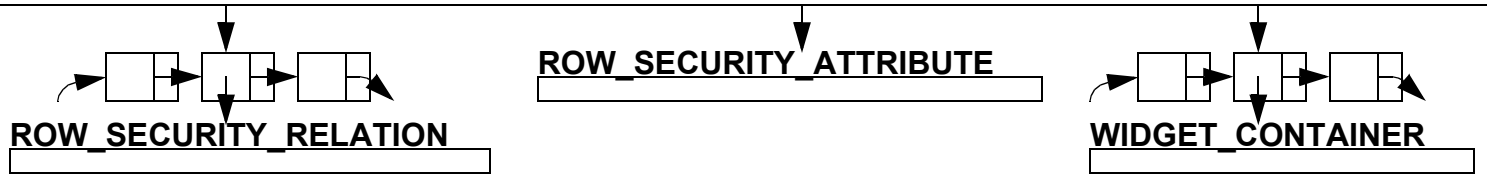
```
/* Usage */
WIDGET_CONTAINER *
   row_security_password_display_container(
      char *attribute_name,
      int attribute_width,
      char *folder_attribute_prompt,
      boolean viewonly_boolean,
      char *widget_password_display_name() );

/* Process */
WIDGET_CONTAINER *widget_container_new(
   widget_password,
   widget_password_display_name
      /* widget_name */ );

widget_container_new()->
   password->
   attribute_name = attribute_name;

widget_container_new()->
   password->
   text->
   attribute_width_max_length =
      attribute_width;

int widget_text_display_size(
   WIDGET_TEXT_DEFAULT_MAX_LENGTH,
   attribute_width );

widget_container_new()->
   password->
   text->
   widget_text_display_size =
      widget_text_display_size();

widget_container_new()->
   password->
   text->
   null_to_slash = 1;

char *row_security_password_display_javascript(
   attribute_name,
   widget_password_display_name );

widget_container_new()->
   password->
   text->
   post_change_javascript =
      row_security_password_display_javascript();

widget_container_new()->
   password->
   text->
   viewonly = viewonly_boolean;

widget_container_new()->heading_string =
   folder_attribute_prompt;

/* Usage */
char *row_security_password_display_javascript(
   char *attribute_name,
   char *widget_password_display_name );

/* Process */
sprintf(
   display_javascript[],
   "timlib_password_set( '%s_$row', '%s_$row' )",
   attribute_name,
   widget_password_display_name );

/* Usage */
WIDGET_CONTAINER *
   row_security_password_checkbox_container(
      char *attribute_name,
      char *widget_password_display_name );

/* Process */
WIDGET_CONTAINER *widget_container_new(
   checkbox,
   (char *)0 /* widget_name */ );

widget_container_new()->
   checkbox->
   prompt = "view";

char *row_security_password_checkbox_javascript(
   attribute_name,
   widget_password_display_name );

widget_container_new()->
   checkbox->
   onclick =
      row_security_password_checkbox_javascript();

/* Usage */
char *row_security_password_checkbox_javascript(
   char *attribute_name,
   char *widget_password_display_name() );

/* Process */
sprintf(
   checkbox_javascript[],
   "timlib_password_view(this,'%s_$row','%s_$row' )",
   attribute_name,
   widget_password_display_name );
```

ROW_SECURITY_VIEWONLY_WIDGET_LIST (1)

```
/* Usage */
ROW_SECURITY_VIEWONLY_WIDGET_LIST *
   row_security_viewonly_widget_list_new(
      char *application_name,
      char *session_key,
      char *login_name,
      LIST *relation_mto1_list,
      LIST *relation_mto1_isa_list,
      LIST *relation_join_one2m_list(),
      LIST *folder_attribute_append_isa_list,
      LIST *no_display_name_list,
      LIST *query_select_name_list,
      LIST *viewonly_attribute_name_list,
      /* Null if not participating */
      ROW_SECURITY_ROLE_UPDATE_LIST *
         row_security_role_update_list );

/* Process */
ROW_SECURITY_VIEWONLY_WIDGET_LIST *
   row_security_viewonly_widget_list_calloc(
      void );

LIST *widget_container_list = list_new();

for FOLDER_ATTRIBUTE *folder_attribute in
   folder_attribute_append_isa_list
{
   if ( folder_attribute->primary_key_index )
   {
      list_set(
         widget_container_list,
         widget_container_new(
            hidden,
            folder_attribute->attribute_name ) );
   }

   if ( !list_string_exists(
         folder_attribute->attribute_name,
         query_select_name_list ) )
   {
      continue;
   }

   if ( list_string_exists(
         folder_attribute->attribute_name,
         no_display_name_list ) )
   {
      continue;
   }

   boolean list_string_exists(
      vewionly_attribute_name_list,
      folder_attribute->attribute_name );

   if (    !list_string_exists()
   &&   ( ROW_SECURITY_RELATION *
         row_security_relation =
            row_security_relation_new(
               (char *)0 /* application_name */,
               (char *)0 /* login_name */,
               (char *)0 /* role_name */,
               (char *)0 /* many_folder_name */,
               (LIST *)0 /* non_primary_name_list */,
               relation_mto1_list
                  /* many_folder_relation_mto1_list */,
               relation_mto1_isa_list,
               folder_attribute->attribute_name,
               folder_attribute->primary_key_index,
               (char *)0 /* post_change_javascript */,
               (DICTIONARY *)0 /* drillthru_dictionary */,
               (SECURITY_ENTITY *)0,
               1 /* viewonly_boolean */,
               0 /* query_row_list_length */,
               0 /* max_query_rows_for_drop_downs */,
               0 /* max_drop_down_size */,
               row_security_relation_list ) ) )
   {
      list_set(
         row_security_relation_list,
         row_security_relation );

      list_set(
         widget_container_list,
            widget_container_new(
               table_data,
               (char *)0 /* widget_name */ ) );

      list_set(
         widget_container_list,
         row_security_relation->
            widget_container );

      continue;
   }
```

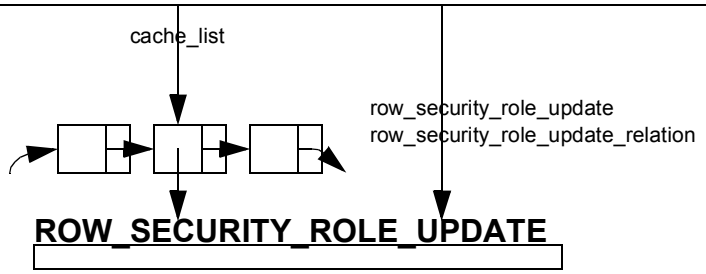
ROW_SECURITY_VIEWONLY_WIDGET_LIST (2)

```
/* Process (continued) */
    list_set(
        viewonly_attribute_name_list,
        folder_attribute->attribute_name );

    if ( ( ROW_SECURITY_ATTRIBUTE *
           row_security_attribute =
            row_security_attribute_new(
               application_name,
               session_key,
               login_name,
               viewonly_attribute_name_list,
               folder_attribute->attribute_name,
               folder_attribute->primary_key_index,
               folder_attribute->prompt,
               folder_attribute->
                  attribute->
                  datatype_name,
               folder_attribute->
                  attribute->
                  width,
               (char *)0 /* post_change_javascript */,
               row_security_relation_list ) ) )
    {
        list_set_list(
            widget_container_list,
            row_security_attribute->
               widget_container_list );
    }
}

if (    row_security_role_update_list
&&  row_security_role_update_list->
        row_security_role_update_relation )
{
    list_set_list(
        widget_container_list,
        LIST *row_security_role_update_container_list(
            row_security_role_update_list->
               attribute_not_null ) );
}

if ( list_length( relation_join_one2m_list ) )
{
    list_set_list(
        widget_container_list,
        LIST *row_security_join_container_list(
            relation_join_one2m_list ) );
}

/* Attributes */
LIST *row_security_relation_list;
ROW_SECURITY_ATTRIBUTE *
    row_security_attribute;
LIST *widget_container_list;
```



ROW_SECURITY_ROLE_UPDATE_LIST (1)

```
/* Usage */
/* Returns null if not participating */
ROW_SECURITY_ROLE_UPDATE_LIST *
   row_security_role_update_list_fetch(
      char *role_name,
      char *folder_name,
      boolean role_override_row_restrictions );

/* Process */
ROW_SECURITY_ROLE_UPDATE_LIST *
   row_security_role_update_list_calloc(
      void );

static LIST *row_security_role_update_list_cache_list(
   role_name );

int list_length(
   row_security_role_update_list_cache_list() );

if ( !list_length() ) return null;

ROW_SECURITY_ROLE_UPDATE *
   row_security_role_update =
      row_security_role_update_seek(
         folder_name,
         row_security_role_update_list_cache_list() );

if ( row_security_role_update )
{
   if (    role_override_row_restrictions
   &&   !row_security_role_update->no_override_boolean )
   {
      return null;
   }

   char *attribute_not_null =
      row_security_role_update->
         attribute_not_null;

   return this;
}

ROW_SECURITY_ROLE_UPDATE *
   row_security_role_update_relation =
      row_security_role_update_relation_seek(
         folder_name,
         row_security_role_update_list_cache_list() );

if ( !row_security_role_update_relation ) return null;

if (    role_override_row_restrictions
&&   !row_security_role_update_relation->
         no_override_boolean )
{
   return null;
}

char *attribute_not_null =
   row_security_role_update_relation->
      attribute_not_null;

char *from =
   row_security_role_update_relation->
      relation_one2m->one_folder_name;

char *row_security_role_update_list_join_where(
   folder_name,
   row_security_role_update_relation->
      relation_one2m->
      one_folder_name,
   row_security_role_update_relation->
      relation_one2m->
      one_folder_primary_key_list,
   row_security_role_update_relation->
      relation_one2m->
      relation_foreign_key_list );

/* Attributes */
LIST *cache_list;
char *role_folder_lookup_in_clause;
ROW_SECURITY_ROLE_UPDATE *
   row_security_role_update;
char *attribute_not_null;
ROW_SECURITY_ROLE_UPDATE *
   row_security_role_update_relation;
char *from;
char *join_where;
```

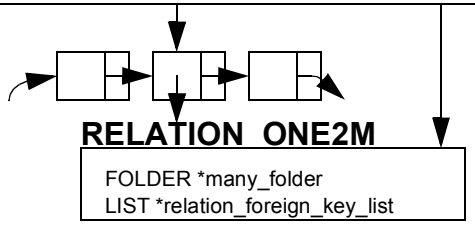


ROW_SECURITY_ROLE_UPDATE_LIST (2)

```
/* Usage */
static LIST *row_security_role_update_list_cache_list(
    char *role_name );

/* Process */
static LIST *cache_list;

if ( cache_list ) return raw_list;

char *role_folder_lookup_in_clause(
    ROLE_FOLDER_TABLE,
    APPASERVER_LOOKUP_STATE,
    APPASERVER_UPDATE_STATE,
    FOLDER_PRIMARY_KEY,
    role_name );

char *appaserver_system_string(
    ROW_SECURITY_ROLE_UPDATE_SELECT,
    ROW_SECURITY_ROLE_UPDATE_TABLE,
    role_folder_lookup_in_clause() );

FILE *appaserver_input_pipe(
    appaserver_system_string() );

for string_input( input[], appaserver_input_pipe() )
{
    list_set(
        list,
        row_security_role_update_parse(
            role_name,
            input ) );
}

pclose( appaserver_input_pipe() );

/* Usage */
char *row_security_role_update_list_join_where(
    char *many_folder_name,
    char *row_security_role_update_relation->
        relation_one2m->
        one_folder_name,
    LIST *row_security_role_update_relation->
        relation_one2m->
        one_folder_primary_key_list,
    LIST *row_security_role_update_relation->
        relation_one2m->
        relation_foreign_key_list );

/* Process */
char *relation_join_where(
    many_folder_name,
    one_folder_name,
    one_folder_primary_key_list,
    relation_foreign_key_list );
```

ROW_SECURITY_ROLE_UPDATE

```
/* Usage */
ROW_SECURITY_ROLE_UPDATE *
   row_security_role_update_parse(
      char *role_name,
      char *input );

/* Process */
ROW_SECURITY_ROLE_UPDATE *
   row_security_role_update_calloc(
      void );

LIST *folder_attribute_fetch_primary_key_list(
   this->folder_name );

LIST *relation_one2m_sans_omit_drillthru_recursive_list(
      (LIST *)0 /* one2m_list Pass in null */,
      this->folder_name
         /* one_folder_name */,
      folder_attribute_fetch_primary_key_list()
         /* one_folder_primary_key_list */ );

/* Usage */
ROW_SECURITY_ROLE_UPDATE *
   row_security_role_update_seek(
      char *folder_name,
      LIST *row_security_role_update_cache_list );

/* Process */
for ROW_SECURITY_ROLE_UPDATE *
   row_security_role_update in
      row_security_role_update_cache_list
{
   if ( strcmp(
         folder_name,
         row_security_role_update->folder_name ) == 0 )
   {
      return row_security_role_update;
   }
}

/* Usage */
ROW_SECURITY_ROLE_UPDATE *
   row_security_role_update_relation_seek(
      char *folder_name,
      LIST *row_security_role_update_cache_list );

/* Process */
for ROW_SECURITY_ROLE_UPDATE *
   row_security_role_update in
      row_security_role_update_cache_list
{
   RELATION_ONE2M *
      row_security_role_update_relation_one2m(
         row_security_role_update->
            relation_one2m_sans_omit_drillthru_recursive_list,
         folder_name );

   if ( row_security_role_update_relation_one2m() )
   {
      return row_security_role_update;
   }
}
return null;

* Usage */
RELATION_ONE2M *
   row_security_role_update_relation_one2m(
      LIST *
         relation_one2m_sans_omit_drillthru_recursive_list,
      char *folder_name );

/* Process */
RELATION_ONE2M *first_relation_one2m = {0};

for RELATION_ONE2M *relation_one2m in
   relation_one2m_sans_omit_drillthru_recursive_list
{
   if ( !first_relation_one2m )
      first_relation_one2m =
         relation_one2m;

   if (   strcmp(
            folder_name,
            relation_one2m->one_folder_name ) == 0 )
   {
      return first_relation_one2m;
   }
}
return null;

/* Attributes */
char *role_name;
char *folder_name;
char *attribute_not_null;
boolean no_override_boolean;
LIST *relation_one2m_sans_omit_drillthru_recursive_list;

/* Set externally */
RELATION_ONE2M *relation_one2m;
```



RELATION (1)

```
/* Usage */
LIST *relation_cache_list(
    void )

/* Process */
static LIST *cache_list = {0};

if ( cache_list ) return cache_list;

cache_list = list_new();

char *relation_system_string(
    const char *RELATION_SELECT,
    const char *RELATION_TABLE,
    char *“pair_one2m_order” /* order_string */ );

FILE *appaserver_input_pipe(
    char *relation_system_string() );

while (
    string_input(
        input[],
        appaserver_input_pipe() )
{
    list_set(
        cache_list,
        RELATION *relation_parse(
            input ) );
}

pclose( appaserver_input_pipe() );

/* Usage */
RELATION *relation_parse(
    char *input );

/* Process */
RELATION *relation_new(
    strdup( many_folder_name ),
    strdup( one_folder_name ),
    strdup( related_attribute_name ) );

/* Usage */
RELATION *relation_new(
    char *many_folder_name,
    char *one_folder_name,
    char *related_attribute_name );

/* Process */
RELATION *relation_calloc(
    void );

/* Attributes */
char *many_folder_name;
char *one_folder_name;
char *related_attribute_name;
int pair_one2m_order;
boolean omit_drillthru;
boolean omit_drilldown;
boolean relation_type_isa;
boolean copy_common_attributes;
boolean automatic_preselection;
boolean drop_down_multi_select;
boolean join_one2m_each_row;
boolean ajax_fill_drop_down;
char *hint_message;
```

RELATION (2)

```
/* Usage */
LIST *relation_seek_mto1_list(
    char *many_folder_name );

/* Process */
LIST *relation_cache_list();

LIST *list = {0};

for RELATION *relation in relation_cache_list()
{
    if ( strcmp(
            relation->many_folder_name,
            many_folder_name ) == 0 )
    {
        if ( !list ) list = list_new();

        list_set( list, relation );
    }
}

/* Usage */
LIST *relation_seek_one2m_list(
    char *one_folder_name );

/* Process */

/* Returns static LIST * */
LIST *relation_cache_list();

LIST *list = {0};

for RELATION *relation in relation_cache_list()
{
    if ( strcmp(
            relation->one_folder_name,
            one_folder_name ) == 0 )
    {
        if ( !list ) list = list_new();

        list_set( list, relation );
    }
}

/* Usage */
LIST *relation_foreign_key_list(
    LIST *one_folder_primary_key_list,
    char *related_attribute_name,
    LIST *foreign_attribute_name_list() );

/* Process */
if ( list_length( foreign_attribute_name_list ) )
{
    LIST *return_list = foreign_attribute_name_list;
}
else
if ( strcmp( related_attribute_name, "null" ) != 0 )
{
    LIST *return_list =
        list_duplicate(
            one_folder_primary_key_list );

    list_replace_last(
        return_list,
        related_attribute_name );
}
else
{
    LIST *return_list =
        list_duplicate(
            one_folder_primary_key_list );
}

/* Usage */
char *relation_name(
    const char ATTRIBUTE_MULTI_KEY_DELIMITER,
    LIST *relation_foreign_key_list() );

/* Process */
list_display(
    relation_foreign_key_list,
    ATTRIBUTE_MULTI_KEY_DELIMITER );

/* Usage */
boolean relation_foreign_key_exists_primary(
    LIST *many_folder_primary_key_list,
    LIST *relation_foreign_key_list() );

/* Process */
for char *foreign_key in
    relation_foreign_key_list
{
    if ( list_string_exists(
        foreign_key,
        many_folder_primary_key_list ) )
    {
        return 1;
    }
}
return 0;
```

RELATION (3)

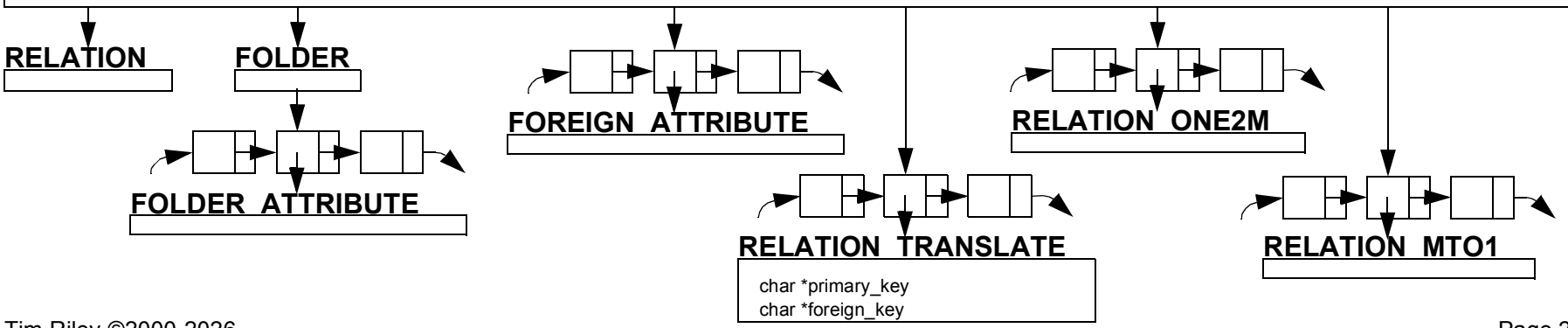
```
/* Usage */
char *relation_prompt(
    char *folder_name,
    char *related_attribute_name,
    LIST *foreign_attribute_name_list(),
    boolean relation_foreign_key_exists_primary() );

/* Process */
char prompt[];
char *ptr = prompt;

if ( relation_foreign_key_exists_primary )
    ptr += sprintf( ptr, "* " );

if ( list_length( foreign_attribute_name_list ) )
{
    ptr += sprintf( ptr,
                "%s",
                string_initial_capital(
                    char buffer[],
                    list_display_delimited(
                        foreign_attribute_name_list,
                        ',' ) ) );
}
else
if ( strcmp( related_attribute_name, "null" ) != 0 )
{
    ptr += sprintf( ptr,
                "%s (%s)",
                string_initial_capital(
                    char buffer1[],
                    folder_name ),
                string_initial_capital(
                    char buffer2[],
                    related_attribute_name ) );
}
else
{
    ptr += sprintf( ptr,
                "%s",
                string_initial_capital(
                    char buffer[],
                    folder_name ) );
}

/* Usage */
char *relation_mnemonic(
    LIST *relation_foreign_key_list() );

/* Process */
char *appaserver_mnemonic(
    relation_foreign_key_list );

/* Usage */
char *relation_insert_statement(
    const char *RELATION_TABLE,
    const char *RELATION_PRIMARY_KEY,
    char *many_folder_name,
    char *one_folder_name,
    char *related_attribute_name );

/* Process */

/* Usage */
char *relation_join_where(
    char *many_folder_name,
    char *one_folder_name,
    LIST *one_folder_primary_key_list,
    LIST *relation_foreign_key_list );

/* Process */
char where[];
char *ptr = where;

strcpy(
    many_table_name[],
    static char *appaserver_table_name(
        many_folder_name ) );

char *one_table_name =
    static char *appaserver_table_name,
        one_folder_name );

list_rewind( one_folder_primary_key_list );

for char *foreign_key in relation_foreign_key_list
{
    if  ( ptr != where ) ptr += sprintf( ptr, " and " );

    char *primary_key =
        list_get(
            one_folder_primary_key_list );

    ptr += sprintf( ptr,
        "%s.%s = %s.%s",
        many_table_name,
        foreign_key,
        one_table_name,
        primary_key );

    list_next( one_folder_primary_key_list );
}
```

RELATION_TRANSLATE

```
/* Usage */
LIST *relation_translate_list(
    LIST *primary_key_list,
    LIST *relation_foreign_key_list,
    char *many_folder_name,
    char *one_folder_name );

/* Process */
LIST *translate_list = list_new();

if (    list_length( primary_key_list ) !=
        list_length( relation_foreign_key_list ) )
{
    appaserver_error_exit(
        many_folder_name,
        one_folder_name );
}

list_rewind( relation_foreign_key_list );

for char *primary_key in primary_key_list
{
    list_set(
        translate_list,
        RELATION_TRANSLATE *
            relation_translate_new(
                primary_key,
                list_get( relation_foreign_key_list )
                    /* foreign_key */ ) );

    list_next( relation_foreign_key_list );
}

/* Usage */
RELATION_TRANSLATE *relation_translate_new(
    char *primary_key,
    char *foreign_key );

/* Process */
RELATION_TRANSLATE *relation_translate_calloc(
    void );

/* Usage */
char *relation_translate_primary_key(
    LIST *relation_translate_list,
    char *foreign_key );

/* Process */
for RELATION_TRANSLATE *relation_translate in
    relation_translate_list
{
    int strcmp(
        relation_translate->foreign_key,
        foreign_key );

    if ( strcmp() == 0 )
    {
        return relation_translate->primary_key;
    }
}
return NULL;

/* Usage */
char *relation_translate_foreign_key(
    LIST *relation_translate_list,
    char *primary_key );

/* Process */
for RELATION_TRANSLATE *relation_translate in
    relation_translate_list
{
    int strcmp(
        relation_translate->primary_key,
        primary_key );

    if ( strcmp() == 0 )
    {
        return relation_translate->foreign_key;
    }
}
return NULL;

/* Usage */
LIST *relation_translate_primary_list(
    LIST *relation_translate_list );

/* Process */

/* Usage */
LIST *relation_translate_foreign_list(
    LIST *relation_translate_list );

/* Process */

/* Usage */
char *relation_translate_primary_delimited_string(
    LIST *relation_translate_list );

/* Process */

/* Usage */
char *relation_translate_foreign_delimited_string(
    LIST *relation_translate_list );

/* Process */

/* Usage */
char *relation_translate_list_display(
    LIST *relation_translate_list );

/* Process */

/* Attibutes */
char *primary_key;
char *foreign_key;
```

RELATION_MTO1 (1)

```
/* Usage */
LIST *relation_mto1_list(
    char *many_folder_name,
    LIST *many_folder_primary_key_list );

/* Process */
LIST *mto1_list = {0};

LIST *relation_seek_mto1_list(
    many_folder_name );

if ( !list_rewind( relation_seek_mto1_list() ) return NULL;

for RELATION *relation in relation_seek_mto1_list()
{
    if ( relation->relation_type_isa ) continue;

    if ( !mto1_list ) mto1_list = list_new();

    RELATION_MTO1 *relation_mto1 =
        relation_mto1_new(
            many_folder_name,
            many_folder_primary_key_list,
            relation );

    list_set( mto1_list, relation_mto1 );
}

/* Usage */
RELATION_MTO1 *relation_mto1_new(
    char *many_folder_name,
    LIST *many_folder_primary_key_list,
    RELATION *relation );

/* Process */
RELATION_MTO1 *relation_mto1_calloc(
    void );

FOLDER *one_folder =
    folder_fetch(
        relation->one_folder_name,
        (LIST *)0 /* exclude_attribute_name_list */,
        /* Sets folder_attribute_primary_list */
        /* Sets folder_attribute_primary_key_list */
        /* Sets folder_attribute_name_list */
        1 /* fetch_folder_attribute_list */,
        1 /* fetch_attribute */,
        1 /* cache_boolean */ );

LIST *foreign_attribute_list(
    many_folder_name,
    relation->one_folder_name,
    relation->related_attribute_name );

if ( list_length( foreign_attribute_list() ) )
{
    LIST *foreign_attribute_name_list(
        foreign_attribute_list() );
}

LIST *relation_foreign_key_list(
    one_folder->primary_key_list,
    relation->related_attribute_name,
    foreign_attribute_name_list() );

LIST *relation_translate_list(
    one_folder->folder_attribute_primary_key_list,
    relation_foreign_key_list(),
    many_folder_name,
    relation->one_folder_name );

char *relation_name(
    ATTRIBUTE_MULTI_KEY_DELIMITER,
    relation_foreign_key_list() );

boolean relation_foreign_key_exists_primary(
    many_folder_primary_key_list,
    relation_foreign_key_list() );

char *relation_prompt(
    relation->one_folder_name,
    relation->related_attribute_name,
    (LIST *)0 /* foreign_attribute_name_list */,
    relation_foreign_key_exists_primary() );

/* Attributes */
char *many_folder_name;
char *one_folder_name;
LIST *many_folder_primary_key_list;
RELATION *relation;
FOLDER *one_folder;
LIST *foreign_attribute_list;
LIST *foreign_attribute_name_list;
LIST *relation_foreign_key_list;
LIST *relation_translate_list;
char *relation_name;
boolean relation_foreign_key_exists_primary;
char *relation_prompt;
LIST *relation_one2m_list;
LIST *relation_mto1_list;
```



RELATION_MTO1 (2)

```
/* Usage */
RELATION_MTO1 *relation_mto1_consumes(
    char *many_attribute_name,
    LIST *relation_mto1_list() );

/* Process */
RELATION_MTO1 *relation_mto1_attribute_consumes(
    many_attribute_name,
    relation_mto1_list );

if ( relation_mto1_attribute_consumes() )
{
    return relation_mto1_attribute_consumes();
}
return NULL;

/* Usage */
RELATION_MTO1 *relation_mto1_isa_consumes(
    char *many_attribute_name,
    LIST *relation_mto1_isa_list() );

/* Process */
for RELATION_MTO1 *relation_mto1 in
    relation_mto1_isa_list
{
    RELATION_MTO1 *relation_mto1_to_one =
        relation_mto1_attribute_consumes(
            many_attribute_name,
            relation_mto1->relation_mto1_list );

    if ( relation_mto1_to_one )
    {
        return relation_mto1_to_one;
    }
}
return NULL;

/* Usage */
RELATION_MTO1 *relation_mto1_attribute_consumes(
    char *attribute_name,
    LIST *relation_mto1_list );

/* Process */
for RELATION_MTO1 *relation_mto1 in
    relation_mto1_list
{
    if ( relation_mto1->relation->omit_drillthru )
            continue;

    boolean list_string_exists(
        attribute_name,
        relation_mto1->relation_foreign_key_list );

    if ( list_string_exists() ) return relation_mto1;
}
return NULL;
```

RELATION_MTO1 (3)

```
/* Usage */
RELATION_MTO1 *relation_mto1_seek(
    char *one_folder_name,
    LIST *relation_mto1_list() );

/* Process */
for RELATION_MTO1 * relation_mto1 in
    relation_mto1_list
{
    if ( strcmp(
         relation_mto1->one_folder_name,
         one_folder_name ) == 0 )
    {
         return relation_mto1;
    }
}
return NULL;

/* Usage */
RELATION_MTO1 *relation_mto1_ajax_fill_seek(
   LIST *relation_mto1_to_one_list );

/* Process */
for RELATION_MTO1 *relation_mto1 in
   relation_mto1_to_one_list
{
   if ( relation_mto1->relation->ajax_fill_drop_down )
        return this;
}
return NULL;

/* Usage */
LIST *relation_mto1_folder_name_list(
   LIST *relation_mto1_list() );

/* Process */

/* Usage */
boolean relation_mto1_exists_multi_select(
   LIST *relation_mto1_list() );

/* Process */

/* Usage */
RELATION_MTO1 *
   relation_mto1_automatic_preselection(
      LIST *relation_mto1_list );

/* Process */
for RELATION_MTO1 *relation_mto1 in
   relation_mto1_list
{
   if ( relation_mto1->
        relation->
        automatic_preselection )
   {
      return relation_mto1;
   }
}
return NULL;

/* Usage */
void relation_mto1_list_set_one_to_many_list(
   LIST *relation_mto1_list /* in/out */ );

/* Process */
for RELATION_MTO1 *relation_mto1 in
   relation_mto1_list
{
   relation_mto1->relation_one2m_list =
      relation_one2m_list(
         relation_mto1->one_folder_name,
         relation_mto1->
            one_folder->
            folder_attribute_primary_key_list,
         0 /* not include_isa_boolean */ );
}
```

RELATION_MTO1 (4)

```
/* Usage */
LIST *relation_mto1_drillthru_list(
    LIST *relation_mto1_list /* Pass in null */,
    char *base_folder_name,
    LIST *base_folder_primary_key_list,
    LIST *fulfilled_folder_name_list );

/* Process */
LIST *relation_seek_mto1_list(
    base_folder_name
        /* many_folder_name */ );

if ( !list_rewind( relation_seek_mto1_list() ) )
    return relation_mto1_list;

for RELATION *relation in
    relation_seek_mto1_list()
{
    if ( list_exists_string(
            relation->one_folder_name,
            fulfilled_folder_name_list ) )
    {
        continue;
    }

    if ( relation->omit_drillthru ) continue;

    RELATION_MTO1 *relation_mto1 =
        relation_mto1_new(
            base_folder_name
                /* many_folder_name */,
            base_folder_primary_key_list
                /* many_folder_primary_key_list */,
            relation );

    if ( !relation_mto1->one_folder->drillthru )
    {
        continue;
    }

    if ( !relation_mto1_list )
    {
        relation_mto1_list = list_new();
    }

    list_set( relation_mto1_list, relation_mto1 );

    relation_mto1_list =
        relation_mto1_drillthru_list(
                relation_mto1_list,
                relation->one_folder_name
                    /* base_folder_name */,
                relation_mto1->
                    one_folder->
                    primary_key_list
                    /* base_folder_primary_key_list */,
                fulfilled_folder_name_list );
}

/* Usage */
LIST *relation_mto1_recursive_list(
    LIST *relation_mto1_list /* Pass in null */,
    char *many_folder_name,
    LIST *many_folder_primary_key_list,
    boolean single_foreign_key_only );

/* Process */
LIST *relation_seek_mto1_list(
    many_folder_name );

if ( !list_rewind( relation_seek_mto1_list() ) )
{
    return relation_mto1_list;
}

for RELATION *relation in
    relation_seek_mto1_list()
{
    if ( relation->relation_type_isa ) continue;

    RELATION_MTO1 *relation_mto1 =
        relation_mto1_new(
            many_folder_name,
            many_folder_primary_key_list,
            relation );

    if ( !relation_mto1_list )
    {
        relation_mto1_list = list_new();
    }

    if ( single_foreign_key_only )
    {
        if ( list_length(
                relation_mto1->
                    relation_foreign_key_list ) == 1 )
        {
            list_set( relation_mto1_list, relation_mto1 );
        }
    }
    else
    {
        list_set( relation_mto1_list, relation_mto1 );
    }

    relation_mto1_list =
        relation_mto1_recursive_list(
            relation_mto1_list,
            relation->one_folder_name
                /* many_folder_name */,
            relation_mto1->
                one_folder->
                primary_key_list
                /* many_folder_primary_key_list */,
            single_foreign_key_only );
}
```

RELATION_MTO1 (5)

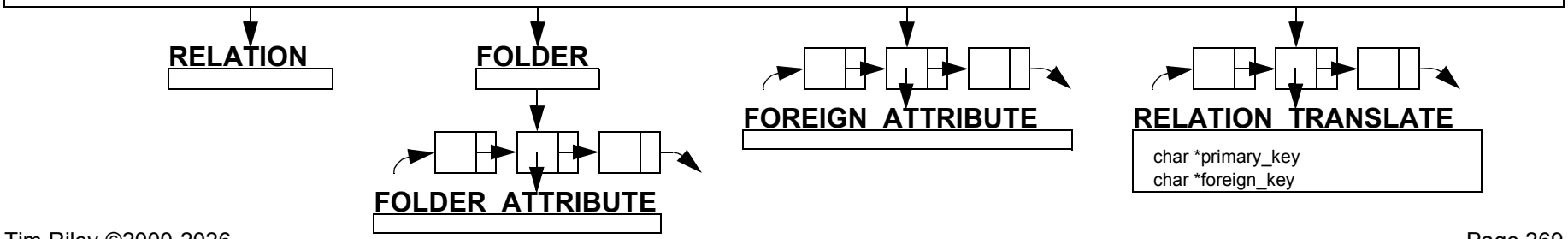
```
/* Usage */
LIST *relation_mto1_isa_list(
    LIST *mto1_isa_list /* Pass in null */,
    char *many_folder_name,
    LIST *many_folder_primary_key_list,
    boolean fetch_relation_one2m_list,
    boolean fetch_relation_mto1_list );

/* Process */
LIST *relation_seek_mto1_list(
    many_folder_name );

if ( !list_rewind( relation_seek_mto1_list() ) )
{
    return mto1_isa_list;
}

for RELATION *relation in
    relation_seek_mto1_list()
{
    if ( !relation->relation_type_isa ) continue;

    RELATION_MTO1 *relation_mto1 =
        relation_mto1_new(
            many_folder_name,
            many_folder_primary_key_list,
            relation );

    if ( !mto1_isa_list ) mto1_isa_list = list_new();

    list_set( mto1_isa_list,relation_mto1 );

    if ( fetch_relation_one2m_list )
    {
        LIST *relation_one2m_list(
            relation_mto1->one_folder_name,
            relation_mto1->
                one_folder->
                folder_attribute_primary_key_list
                /* one_folder_primary_key_list */,
            0 /* not include_isa_boolean */ );
    }

    if ( fetch_relation_mto1_list )
    {
        LIST *relation_mto1_list(
            many_folder_name,
            many_folder_primary_key_list );
    }

    mto1_isa_list =
        relation_mto1_isa_list(
            mto1_isa_list,
            relation->one_folder_name
                /* many_folder_name */,
            relation_mto1->
                one_folder->
                primary_key_list
                /* many_folder_primary_key_list */,
            fetch_relation_one2m_list,
            fetch_relation_mto1_list );
}

/* Usage */
char *relation_mto1_where_string(
    const char *FOLDER_PRIMARY_KEY,
    char *many_folder_name );

/* Process */
snprintf(
    static where_string[],
    sizeof ( where_string ),
    "%s = '%s'",
    FOLDER_PRIMARY_KEY,
    many_folder_name );

/* Usage */
char *relation_mto1_list_display(
    LIST *relation_mto1_list() );

/* Process */

/* Usage */
LIST *relation_mto1_to_one_fetch_list(
    char *one_folder_name,
    LIST *one_folder_primary_key_list );

/* Process */
LIST *relation_mto1_list(
    one_folder_name
        /* many_folder_name */,
    one_folder_primary_key_list
        /* many_folder_primary_key_list */ );

/* Usage */
LIST *relation_mto1_omit_drillthru_list(
    LIST *relation_mto1_list );

/* Process */

/* Usage */
LIST *relation_mto1_without_omit_drillthru_list(
    LIST *relation_mto1_list );

/* Process */

/* Usage */
LIST *relation_mto1_to_one_list(
    LIST *relation_mto1_list,
    LIST *relation_mto1_isa_list );

/* Process */
LIST *mto1_list = list_new();

for RELATION_MTO1 *relation_mto1 in
    relation_mto1_list
{
    list_set( mto1_list, relation_mto1 );
}

for RELATION_MTO1 *relation_mto1 in
    relation_mto1_isa_list
{
    list_set_list(
        mto1_list,
        relation_mto1->relation_mto1_list );
}
```

RELATION_MTO1 (6)

```
/* Usage */
LIST *relation_mto1_common_name_list(
    LIST *folder_attribute_non_primary_name_list,
    boolean relation_mto1->
        relation->
        copy_common_columns,
    LIST *one_folder_attribute_name_list );

/* Process */

/* Usage */
LIST *relation_mto1_foreign_key_less_equal_list(
    int QUERY_DROP_DOWN_FETCH_MAX_ROWS,
    int DRILLTHRU_SKIPPED_MAX_FOREIGN_LENGTH,
    LIST *relation_mto1_list );

/* Process */
for RELATION_MTO1 *relation_mto1 in relation_mto1_list
{
    int list_length( relation_mto1->relation_foreign_key_list );

    if (    list_length() <=
            DRILLTHRU_SKIPPED_MAX_FOREIGN_LENGTH )
    {
        unsigned long query_row_count(
            relation_mto1->one_folder_name );

        if (    query_row_count() <=
                QUERY_DROP_DOWN_FETCH_MAX_ROWS )
        {
            list_set( LIST *list, relation_mto1 );
        }
    }
}
```

RELATION_ONE2M (1)

```
/* Usage */
LIST *relation_one2m_list(
    char *one_folder_name,
    LIST *one_folder_primary_key_list,
    boolean include_isa_boolean );

/* Process */
LIST *one2m_list = {0};

LIST *relation_seek_one2m_list(
    one_folder_name );

if ( !list_rewind( relation_seek_one2m_list() ) )
        return NULL;

for RELATION *relation in relation_seek_one2m_list()
{
    if (    !include_isa_boolean
    &&  relation->relation_type_isa )
    {
        continue;
    }

    RELATION_ONE2M *relation_one2m =
        relation_one2m_new(
            one_folder_name,
            one_folder_primary_key_list,
            relation );

    if ( !one2m_list ) one2m_list = list_new();

    list_set( one2m_list, relation_one2m );
}

/* Usage */
RELATION_ONE2M *relation_one2m_new(
    char *one_folder_name,
    LIST *one_folder_primary_key_list,
    RELATION *relation );

/* Process */
RELATION_ONE2M *relation_one2m_calloc(
    void );

FOLDER *many_folder =
    folder_fetch(
        relation->many_folder_name,
        (LIST *)0 /* exclude_attribute_name_list */,
        /* Sets folder_attribute_primary_list */
        /* Sets folder_attribute_primary_key_list */
        /* Sets folder_attribute_name_list */
        1 /* fetch_folder_attribute_list */,
        1 /* fetch_attribute */,
        1 /* cache_boolean */ );

LIST *foreign_attribute_list(
    relation->many_folder_name,
    one_folder_name,
    relation->related_attribute_name );

if ( list_length( foreign_attribute_list() ) )
{
    LIST *foreign_attribute_name_list(
        foreign_attribute_list() );
}

LIST *relation_foreign_key_list(
    one_folder_primary_key_list,
    relation->related_attribute_name,
    foreign_attribute_name_list() );

LIST *relation_translate_list(
    one_folder_primary_key_list,
    relation_foreign_key_list,
    relation->many_folder_name,
    one_folder_name );

char *relation_name(
    ATTRIBUTE_MULTI_KEY_DELIMITER,
    relation_foreign_key_list() );

boolean relation_foreign_key_exists_primary(
    many_folder->primary_key_list,
    relation_foreign_key_list() );

char *relation_prompt(
    relation->many_folder_name,
    relation->related_attribute_name,
    foreign_attribute_name_list(),
    relation_foreign_key_exists_primary() );

boolean relation_one2m_primary_key_subset(
    relation_foreign_key_list(),
    many_folder->folder_attribute_primary_key_list );

boolean relation_one2m_foreign_key_none_primary(
    relation_foreign_key_list(),
    many_folder->folder_attribute_primary_key_list );

/* Attributes */
char *one_folder_name;
char *many_folder_name;
LIST *one_folder_primary_key_list;
RELATION *relation;
FOLDER *many_folder;
LIST *foreign_attribute_list;
LIST *foreign_attribute_name_list;
LIST *relation_foreign_key_list;
LIST *relation_translate_list;
char *relation_name;
boolean relation_foreign_key_exists_primary;
char *relation_prompt;
boolean primary_key_subset;
boolean foreign_key_none_primary;
```



RELATION_ONE2M (2)

```
/* Usage */
LIST *relation_one2m_recursive_list(
    LIST *one2m_list /* Pass in NULL */,
    char *one_folder_name,
    LIST *one_folder_primary_key_list );

/* Process */
LIST *relation_seek_one2m_list(
    char *one_folder_name );

if ( !list_rewind( relation_seek_one2m_list() )
    return one2m_list;

for RELATION *relation in
    relation_seek_one2m_list()
{
    RELATION_ONE2M *relation_one2m =
        relation_one2m_new(
            one_folder_name,
            one_folder_primary_key_list,
            relation );

    if ( !one2m_list ) one2m_list = list_new();

    list_set( one2m_list, relation_one2m );

    if ( relation_one2m->primary_key_subset )
    {
        one2m_list =
            relation_one2m_recursive_list(
                one2m_list,
                relation->many_folder_name
                    /* one_folder_name */,
                relation_one2m->
                    many_folder->
                    primary_key_list
                    /* one_folder_primary_key_list */ );
    }
}

/* Usage */
LIST *relation_one2m_sans_omit_drillthru_recursive_list(
    LIST *one2m_list /* Pass in NULL */,
    char *one_folder_name,
    LIST *one_folder_primary_key_list );

/* Process */
LIST *relation_seek_one2m_list(
    char *one_folder_name );

if ( !list_rewind( relation_seek_one2m_list() )
    return one2m_list;

for RELATION *relation in
    relation_seek_one2m_list()
{
    if ( relation->omit_drillthru ) continue;

    RELATION_ONE2M *relation_one2m =
        relation_one2m_new(
            one_folder_name,
            one_folder_primary_key_list,
            relation );

    if ( !one2m_list ) one2m_list = list_new();

    list_set( one2m_list, relation_one2m );

    if ( relation_one2m->primary_key_subset )
    {
        one2m_list =
            relation_one2m_recursive_list(
                one2m_list,
                relation->many_folder_name
                    /* one_folder_name */,
                relation_one2m->
                    many_folder->
                    primary_key_list
                    /* one_folder_primary_key_list */ );
    }
}

/* Usage */
boolean relation_one2m_primary_key_subset(
    LIST *relation_foreign_key_list,
    LIST *many_folder_primary_key_list );

/* Process */
return
list_is_subset_of(
    relation_foreign_key_list /* subset */,
    many_folder_primary_key_list /* set */ );

/* Usage */
boolean relation_one2m_foreign_key_none_primary(
    LIST *relation_foreign_key_list,
    LIST *many_folder_primary_key_list );

/* Process */
for char *foreign_key in relation_foreign_key_list
{
    boolean
    list_string_exists(
        foreign_key,
        many_folder_primary_key_list );

    if ( list_string_exists() ) return 0;
}
return 1;
```

RELATION_ONE2M (3)

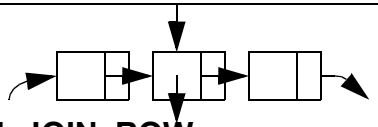
```
/* Usage */
LIST *relation_one2m_pair_list(
    char *one_folder_name,
    LIST *one_folder_primary_key_list );

/* Process */
LIST *relation_seek_one2m_list(
    char *one_folder_name );

if ( !list_rewind( relation_seek_one2m_list() )
    return one2m_list;

for RELATION *relation in
    relation_seek_one2m_list()
{
    if ( !relation->pair_one2m_order ) continue;

    RELATION_ONE2M *relation_one2m =
        relation_one2m_new(
            one_folder_name,
            one_folder_primary_key_list,
            relation );

    if ( !one2m_list ) one2m_list = list_new();

    list_set( one2m_list, relation_one2m );
}

/* Usage */
LIST *relation_one2m_folder_name_list(
    LIST *relation_one2m_list() );

/* Process */
for RELATION_ONE2M *relation_one2m in
    relation_one2m_list
{
    list_set( list, relation_one2m->many_folder_name );
}

/* Usage */
LIST *relation_one2m_join_list(
    char *one_folder_name,
    LIST *one_folder_primary_key_list,
    DICTIONARY *no_display_dictionary );

/* Process */
LIST *relation_seek_one2m_list(
    char *one_folder_name );

if ( !list_rewind( relation_seek_one2m_list() )
    return one2m_list;

for RELATION *relation in
    relation_seek_one2m_list()
{
    if ( !relation->join_one2m_each_row ) continue;

    if ( dictionary_key_exists(
            no_display_dictionary,
            relation->many_folder_name ) )
    {
        continue;
    }

    RELATION_ONE2M *relation_one2m =
        relation_one2m_new(
            one_folder_name,
            one_folder_primary_key_list,
            relation );

    if ( !one2m_list ) one2m_list = list_new();

    list_set( one2m_list, relation_one2m );
}

/* Usage */
char *relation_one2m_where_string(
    const char *RELATION_FOREIGN_KEY
        /* probably related_table */,
    char *one_folder_name );

/* Process */
snprintf(
    static where_string[],
    sizeof ( where_string ),
    "%s = '%s'",
    RELATION_FOREIGN_KEY,
    one_folder_name );

/* Usage */
char *relation_one2m_list_display(
    LIST *relation_one2m_list );

/* Process */
```

RELATION_JOIN

```
/* Usage */
RELATION_JOIN *relation_join_new(
    LIST *one_folder_primary_key_list,
    LIST *relation_one2m_join_list(),
    LIST *query_fetch_row_list  );

/* Process */
RELATION_JOIN *relation_join_calloc(
    void );
```

```
LIST *relation_join_row_list = list_new();

for QUERY_ROW *query_row in
    query_fetch_row_list
{
    RELATION_JOIN_ROW *relation_join_row =
        relation_join_row_new(
            one_folder_primary_key_list,
            relation_one2m_join_list,
            query_row->cell_list );
```

```
        list_set(
            relation_join_row_list,
            relation_join_row );
}

/* Attribute */
LIST *relation_join_row_list;
```

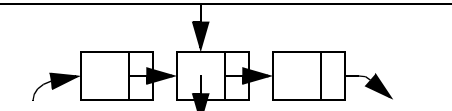


RELATION_JOIN_ROW

```
/* Usage */
RELATION_JOIN_ROW *relation_join_row_new(
    LIST *one_folder_primary_key_list,
    LIST *relation_one2m_join_list(),
    LIST *query_row_cell_list  );

/* Process */
RELATION_JOIN_ROW *relation_join_row_calloc( void );

LIST *relation_join_folder_list = list_new();

for RELATION_ONE2M *relation_one2m_join in
    relation_one2m_join_list
{
    RELATION_JOIN_FOLDER *relation_join_folder =
        relation_join_folder_new(
            one_folder_primary_key_list,
            relation_one2m_join,
```

```
            query_row_cell_list );

    if ( relation_join_folder )
    {
        list_set(
            relation_join_folder_list,
            relation_join_folder );
    }
}

/* Usage */
LIST *relation_join_row_folder_name_list(
    RELATION_JOIN_ROW *relation_join_row );

/* Process */
static LIST *folder_name_list = {0};

if ( !relation_join_row ) return NULL;
```

```
if ( folder_name_list ) return folder_name_list;

folder_name_list = list_new();

for RELATION_JOIN_FOLDER *relation_join_folder in
    relation_join_row->relation_join_folder_list
{
    list_set(
        folder_name_list,
        relation_join_folder->
            relation_one2m_join->
            many_folder_name );
}

/* Attribute */
LIST *relation_join_folder_list;
```



RELATION_JOIN_FOLDER

RELATION_JOIN_FOLDER

```
/* Usage */
RELATION_JOIN_FOLDER *relation_join_folder_new(
    LIST *one_folder_primary_key_list,
    RELATION_ONE2M *relation_one2m_join,
    LIST *query_row_cell_list );

/* Process */
RELATION_JOIN_FOLDER *relation_join_folder_calloc(
    void );

LIST *relation_join_folder_select_name_list(
    relation_one2m_join->
        many_folder->
        folder_attribute_primary_key_list,
    relation_one2m_join->
        relation_foreign_key_list );

char *relation_join_folder_select_name_list_string(
    relation_join_folder_select_name_list() );

RELATION_JOIN_FOLDER_WHERE *
    relation_join_folder_where =
        relation_join_folder_where_new(
            one_folder_primary_key_list,
            relation_one2m_join->
                relation_foreign_key_list,
            query_row_cell_list );

if ( !relation_join_folder_where ) return NULL;

char *relation_join_folder_system_string(
    relation_join_folder_select_name_list_string(),
    relation_one2m_join->many_folder_name,
    relation_join_folder_where->string );

LIST *relation_join_folder_delimited_list(
    relation_join_folder_system_string() );
```

```
/* Usage */
LIST *relation_join_folder_select_name_list(
    LIST *many_folder_primary_key_list,
    LIST *relation_foreign_key_list );

/* Process */
LIST *list_subtract(
    many_folder_primary_key_list,
    relation_foreign_key_list );

/* Usage */
char *relation_join_folder_select_name_list_string(
    LIST *relation_join_folder_select_name_list() );

/* Process */
char *list_delimited(
    relation_join_folder_select_name_list,
    ',' );

/* Usage */
char *relation_join_folder_system_string(
    char *relation_join_folder_select_name_list_string(),
    char *many_folder_name,
    char *relation_join_folder_where_string );

/* Process */
char *appaserver_table_name(
    many_folder_name );

snprintf( system_string[],
    "select.sh \"%s\" %s \"%s\" select",
    sizeof ( system_string ),
    relation_join_folder_select_name_list_string,
    appaserver_table_name(),
    relation_join_folder_where_string );
```

```
/* Usage */
LIST *relation_join_folder_delimited_list(
    char *relation_join_folder_system_string() );

/* Process */
LIST *list_pipe_fetch( system_string );

/* Usage */
RELATION_JOIN_FOLDER *relation_join_folder_seek(
    char *folder_name,
    LIST *relation_join_folder_list );

/* Process */
for RELATION_JOIN_FOLDER *relation_join_folder in
    relation_join_folder_list
{
    if ( strcmp(
            relation_join_folder->
                relation_one2m_join->
                many_folder_name,
            folder_name ) == 0 )
    {
        return relation_join_folder;
    }
}
return NULL;

/* Attributes */
RELATION_ONE2M *relation_one2m_join;
LIST *select_name_list;
char *select_name_list_string;
RELATION_JOIN_FOLDER_WHERE *
    relation_join_folder_where;
char *system_string;
LIST *delimited_list;
```

RELATION_JOIN_FOLDER_WHERE

```
RELATION_JOIN_FOLDER_WHERE *
    relation_join_folder_where_new(
        LIST *one_folder_primary_key_list,
        LIST *relation_one2m_join->
            relation_foreign_key_list,
        LIST *query_row_cell_list );

/* Process */
RELATION_JOIN_FOLDER_WHERE *
    relation_join_folder_where_calloc(
        void );

LIST *query_cell_attribute_data_list(
    one_folder_primary_key_list
        /* attribute_name_list */,
    query_row_cell_list );

if ( !query_cell_attribute_data_list() ) return NULL;
```

```
char *relation_join_folder_where_string(
    relation_foreign_key_list,
    query_cell_attribute_data_list() );

/* Usage */
char *relation_join_folder_where_string(
    LIST *relation_foreign_key_list,
    LIST *query_cell_attribute_data_list() );

/* Process */
char where_string[];
char *ptr = where_string;

list_rewind( relation_foreign_key_list );

for char *attribute_datum in
    query_cell_attribute_data_list
{
    if ( ptr != where_string )
        ptr += sprintf( ptr, " and " );

    char *security_sql_injection_escape(
        SECURITY_ESCAPE_CHARACTER_STRING,
        attribute_datum );

    ptr += sprintf( ptr,
        "%s = '%s'",
        (char *)list_get( relation_foreign_key_list ),
        security_sql_injection_escape() );

    list_next( relation_foreign_key_list );
}

/* Attributes */
LIST *query_cell_attribute_data_list;
char *string;
```

RELATION_COPY (1)

```
/* Usage */
RELATION_COPY *relation_copy_new(
    DICTIONARY *dictionary /* in/out */,
    LIST *folder_attribute_list,
    LIST *relation_mto1_list,
    int row_number );

/* Process */
RELATION_COPY *relation_copy_calloc(
    void );

for RELATION_MTO1 *relation_mto1 in
    relation_mto1_list
{
    if ( relation_mto1->
            relation->
            copy_common_columns )
    {
        if ( !folder_attribute_non_primary_name_list() )
        {
            folder_attribute_non_primary_name_list(
                folder_attribute_list );
        }

        void relation_copy_set(
            dictionary /* in/out */,
            folder_attribute_non_primary_name_list(),
            relation_mto1,
            row_number );
    }
}
```

```
/* Usage */
void relation_copy_set(
    DICTIONARY *dictionary /* in/out */,
    LIST *folder_attribute_non_primary_name_list,
    RELATION_MTO1 *relation_mto1,
    int row_number );

/* Process */
LIST *relation_mto1_common_name_list(
    folder_attribute_non_primary_name_list,
    1 /* copy_common_columns */,
    relation_mto1->
        one_folder->
        folder_attribute_name_list );

LIST *relation_copy_where_query_cell_list(
    dictionary,
    relation_mto1->relation_foreign_key_list,
    relation_mto1->
        one_folder->
        folder_attribute_primary_key_list,
    row_number );

if ( !relation_copy_where_query_cell_list() ) return;

char *query_cell_where_string(
    insert_copy_where_query_cell_list() );

char *relation_copy_system_string(
    ',' /* delimiter */,
    relation_mto1_common_name_list(),
        /* select_name_list */,
    relation_mto1->one_folder_name,
    query_cell_where_string() );

QUERY_FETCH *query_fetch_new(
    relation_mto1->one_folder_name,
    (LIST *)0 /* folder_attribute_list */,
    date_convert_blank,
    relation_mto1_common_name_list()
        /* query_select_name_list */,
    relation_copy_system_string()
        /* query_system_string */,
    0 /* not input_save_boolean */ );

LIST *relation_copy_result_query_cell_list(
    query_fetch_new()->row_list );

for QUERY_CELL *query_cell in
    insert_copy_result_query_cell_list()
{
    char *relation_copy_attribute_name(
        query_cell->attribute_name,
        row_number );

    dictionary_set(
        dictionary,
        relation_copy_attribute_name(),
        query_cell->select_datum );
}

/* Attributes */
LIST *folder_attribute_non_primary_name_list;
```

RELATION_COPY (2)

```
/* Usage */
LIST *relation_copy_where_query_cell_list(
    DICTIONARY *dictionary,
    LIST *relation_foreign_key_list,
    LIST *folder_attribute_primary_key_list,
    int row_number );

/* Process */
LIST *query_cell_list = list_new();

list_rewind( relation_foreign_key_list );

for char *primary_key in
    folder_attribute_primary_key_list
{
    char *dictionary_index_get(
        (char *)list_get( relation_foreign_key_list ),
        dictionary,
        row_number /* index */ );

    if ( !dictionary_index_get() ) return null;

    QUERY_CELL *query_cell_simple_new(
        primary_key /* attribute_name */,
        dictionary_get() /* select_datum */ );
    list_set( query_cell_list, query_cell_simple_new() );
    list_next( relation_foreign_key_list );
}

/* Usage */
char *relation_copy_system_string(
    const char delimiter,
    LIST *select_name_list,
    char *folder_name,
    char *where_string );

/* Process */
char *list_display_delimited(
    select_name_list,
    delimiter ),

char *appaserver_table_name( folder_name );

snprintf(
    system_string[],
    sizeof ( system_string ),
    "select.sh %s %s \"%s\"",
    list_display_delimited(),
    appaserver_table_name(),
    where_string );
/* Usage */
LIST *relation_copy_result_query_cell_list(
    LIST *query_fetch_row_list );

/* Process */
QUERY_ROW *query_row =
    list_first(
        query_fetch_row_list );

query_row->cell_list;

/* Usage */
char *relation_copy_attribute_name(
    char *query_cell_attribute_name,
    int row_number );

/* Process */
if ( !row_number ) return query_cell_attribute_name;

char *dictionary_attribute_name_append_row_number(
    query_cell_attribute_name,
    row_number );

char *strdup(
    dictionary_attribute_name_append_row_number() );
```

FOLDER (1)

```
/* Usage */
FOLDER *folder_fetch(
    char *folder_name,
    LIST *role_attribute_exclude_lookup_name_list,
    /* Sets folder_attribute_primary_list */
    /* Sets folder_attribute_primary_key_list */
    /* Sets folder_attribute_name_list */
    boolean fetch_folder_attribute_list,
    boolean fetch_attribute,
    boolean cache_boolean );

/* Process */
if ( cache_boolean )
{
    FOLDER *folder =
        folder_cache_fetch(
            folder_name,
            role_attribute_exclude_lookup_name_list,
            fetch_folder_attribute_list,
            fetch_attribute );
}

if ( !folder )
{
    FOLDER *folder =
        folder_where_fetch(
            folder_name,
            role_attribute_exclude_lookup_name_list,
            fetch_folder_attribute_list,
            fetch_attribute );
}

if ( !folder ) exit( 1 );

/* Usage */
FOLDER *folder_cache_fetch(
    char *folder_name,
    LIST *role_attribute_exclude_lookup_name_list,
    boolean fetch_folder_attribute_list,
    boolean fetch_attribute );

/* Process */
static LIST *list = {0};

if ( !list )
{
    LIST *list =
        folder_list(
            role_attribute_exclude_lookup_name_list,
            fetch_folder_attribute_list,
            fetch_attribute );
}

FOLDER *folder_seek(
    char *folder_name,
    LIST *list /* folder_list */ );

/* Usage */
FOLDER *folder_where_fetch(
    char *folder_name,
    LIST *role_attribute_exclude_lookup_name_list,
    boolean fetch_folder_attribute_list,
    boolean fetch_attribute );

/* Process */
char *folder_primary_where_string(
    const char *FOLDER_PRIMARY_KEY,
    char *folder_name );

char *appaserver_system_string(
    FOLDER_SELECT,
    FOLDER_TABLE,
    folder_primary_where_string() );

char *string_pipe_input(
    appaserver_system_string() );

FOLDER *folder_parse(
    role_attribute_exclude_lookup_name_list,
    fetch_folder_attribute_list,
    fetch_attribute,
    0 /* not cache_boolean */,
    string_pipe_input() );

/* Usage */
FOLDER *folder_parse(
    LIST *role_attribute_exclude_lookup_name_list,
    boolean fetch_folder_attribute_list,
    boolean fetch_attribute,
    boolean cache_boolean,
    char *input );

/* Process */
FOLDER *folder_new( strdup( folder_name ) );

if ( fetch_folder_attribute_list )
{
    LIST *folder_attribute_list(
        folder_name,
        role_attribute_exclude_lookup_name_list,
        fetch_attribute,
        cache_boolean  );

    LIST *folder_attribute_primary_list(
        folder_name,
        folder_attribute_list() );

    LIST *folder_attribute_primary_key_list(
        this->folder_name,
        folder_attribute_primary_list() );

    LIST *folder_attribute_name_list(
        folder_name,
        folder_attribute_list() );
}

/* Usage */
FOLDER *folder_new(
     char *folder_name );

/* Process */
FOLDER *folder_calloc(
     void );
```

FOLDER (2)

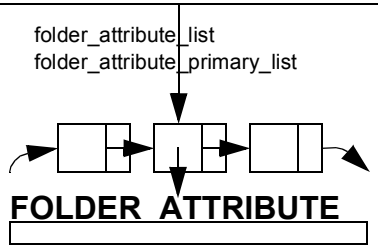
```
/* Usage */
LIST *folder_list(
    LIST *role_attribute_exclude_lookup_name_list,
    boolean fetch_folder_attribute_list,
    boolean fetch_attribute );

/* Process */
char *appaserver_system_string(
    FOLDER_SELECT,
    FOLDER_TABLE,
    (char *)0 /* where_string */ );

FILE *appaserver_input_pipe(
    char *appaserver_system_string() );

LIST *list = list_new();

while ( string_input( input[], appaserver_input_pipe() )
{
    list_set(
        list,
        folder_parse(
            role_attribute_exclude_lookup_name_list,
            fetch_folder_attribute_list,
            fetch_attribute,
            1 /* cache_boolean */,
            string_input() );
}
pclose( appaserver_input_pipe() );

/* Usage */
LIST *folder_table_name_list(
      char *role_name );
```

```
/* Process */
LIST *name_list = list_new();

char *folder_role_where_string(
      ROLE_FOLDER_TABLE,
      FOLDER_PRIMARY_KEY,
      role_name );

char *appaserver_system_string(
    FOLDER_PRIMARY_KEY /* select */,
    FOLDER_TABLE,
    folder_role_where_string() );

FILE *appaserver_input_pipe(
    appaserver_system_string() );

for char *folder_name in appaserver_input_pipe()
{
    char *application_table_name( folder_name );

    list_set(
        name_list,
        strdup( application_table_name() );
}
pclose( appaserver_input_pipe() );

/* Usage */
char *folder_role_where_string(
    const char *ROLE_FOLDER_TABLE,
    const char *FOLDER_PRIMARY_KEY,
    char *role_name_list_string );
```

```
/* Process */
LIST *string_list( role_name_list_string );

char *string_in_clause(
    string_list() );

sprintf(
        static where_string[],
        "%s in (select %s from %s where "
        "role in (%s) )",
        FOLDER_PRIMARY_KEY,
        FOLDER_PRIMARY_KEY,
        ROLE_FOLDER_TABLE,
        string_in_clause() );

/* Usage */
unsigned long folder_row_count(
      char *appaserver_table_name() );

/* Process */
sprintf( system_string[],
    "echo \"select count(1) from %s;\" |"
    "sql_timeout.sh 1",
    appaserver_table_name );

char *string_pipe_fetch( system_string );

if ( string_pipe_fetch() )
    return strtoul( string_pipe_fetch(), NULL, 10 );
else
    return 0;
```

FOLDER (3)

```
/* Usage */
char *folder_form(
    char *folder_name );

/* Process */
FOLDER *folder_fetch(
    folder_name,
    (LIST *)0 /* exclude_attribute_name_list */,
    0 /* not fetch_folder_attribute_list */,
    0 /* not fetch_attribute */,
    0 /* not cache_boolean */ );

return folder_fetch()->folder_form;

/* Usage */
char *folder_notepad(
    char *folder_name );

/* Process */
FOLDER *folder_fetch(
    folder_name,
    (LIST *)0 /* exclude_attribute_name_list */,
    0 /* not fetch_folder_attribute_list */,
    0 /* not fetch_attribute */,
    0 /* not cache_boolean */ );

return folder_fetch()->notepad;

/* Usage */
boolean folder_no_initial_capital(
    char *folder_name );

/* Process */
if (    !folder_name
||      strcmp( folder_name, "null" ) == 0 )
{
    return 0;
}

FOLDER *folder_fetch(
    folder_name,
    (LIST *)0 /* exclude_attribute_name_list */,
    0 /* not fetch_folder_attribute_list */,
    0 /* not fetch_attribute */,
    0 /* not cache_boolean */ );

return folder_fetch()->no_initial_capital;

/* Usage */
LIST *folder_name_list(
    LIST *folder_list() );

/* Process */

/* Usage */
LIST *folder_system_name_list(
    const char *ROLE_SYSTEM );

/* Process */

/* Usage */
LIST *folder_fetch_name_list(
    const char *FOLDER_PRIMARY_KEY,
    const char *FOLDER_TABLE );

/* Process */

/* Usage */
boolean folder_form_prompt_boolean(
    const char *FORM_APPASERVER_PROMPT,
    char *appaserver_form );

/* Process */

/* Usage */
char *folder_column_fetch(
    const char *column_name,
    char *folder_name );

/* Process */
char *folder_primary_where_string(
    FOLDER_PRIMARY_KEY,
    folder_name );

char *appaserver_system_string(
    column_name /* select */,
    FOLDER_TABLE,
    folder_primary_where_string() );

char *string_pipe_input(
    appaserver_system_string() );

/* Attributes */
char *folder_name;
char *folder_form;
int insert_rows_number;
char *notepad;
char *populate_drop_down_process_name;
char *post_change_process_name;
char *html_help_file_anchor;
char *post_change_javascript;
char *subschema;
char *storage_engine;
boolean exclude_application_export;
boolean drillthru;
boolean no_initial_capital;
char *index_directory;
char *data_directory;
char *create_view_statement;
char *javascript_filename;
LIST *folder_attribute_list;
LIST *folder_attribute_primary_list;
LIST *primary_key_list;
LIST *folder_attribute_name_list;

/* Set externally */
PROCESS *post_change_process;
```

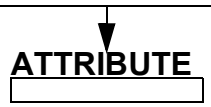


FOLDER_ROW_LEVEL_RESTRICTION

```
/* Usage */
FOLDER_ROW_LEVEL_RESTRICTION *
    folder_row_level_restriction_fetch(
        char *folder_name );

/* Process */
char *folder_primary_where_string(
    FOLDER_PRIMARY_KEY,
    folder_name );

char *appaserver_system_string(
    char *FOLDER_ROW_LEVEL_RESTRICTION_SELECT,
    char *FOLDER_ROW_LEVEL_RESTRICTION_TABLE,
    char *folder_primary_where_string() );

char *string_pipe_input(
    appaserver_system_string() );

FOLDER_ROW_LEVEL_RESTRICTION *
    folder_row_level_restriction_parse(
        string_pipe_input() );

/* Usage */
FOLDER_ROW_LEVEL_RESTRICTION *
    folder_row_level_restriction_parse(
        char *input );

/* Process */
FOLDER_ROW_LEVEL_RESTRICTION *
    folder_row_level_restriction_calloc(
        void );

boolean
    folder_row_level_restriction_non_owner_viewonly(
    char *input );

boolean
    folder_row_level_restriction_non_owner_forbid(
    char *input );

/* Usage */
char *folder_row_level_restriction_string(
    char *folder_name );

/* Process */
char *folder_primary_where_string(
    FOLDER_PRIMARY_KEY,
    folder_name );

sprintf(
    system_string[],
    "select.sh %s %s \"%s\"",
    "row_level_restriction",
    FOLDER_ROW_LEVEL_RESTRICTION_TABLE,
    folder_primary_where_string() );

char *string_pipe_fetch( system_string );

/* Attributes */
boolean non_owner_viewonly;
boolean non_owner_forbid;
```

FOLDER_ATTRIBUTE (1)

```
/* Usage */
LIST *folder_attribute_fetch_list(
    char *folder_name,
    LIST *role_attribute_exclude_lookup_name_list,
    boolean fetch_attribute );

/* Process */
char *folder_attribute_fetch_where_string(
    FOLDER_PRIMARY_KEY,
    folder_name,
    role_attribute_exclude_lookup_name_list );

char *folder_attribute_system_string(
    FOLDER_ATTRIBUTE_SELECT,
    FOLDER_ATTRIBUTE_TABLE,
    folder_attribute_fetch_where_string() );

LIST *folder_attribute_system_list(
    folder_attribute_system_string(),
    fetch_attribute );

/* Usage */
FOLDER_ATTRIBUTE *folder_attribute_parse(
    char *input,
    boolean fetch_attribute );

/* Process */
FOLDER_ATTRIBUTE *folder_attribute_new(
    folder_name,
    attribute_name );

if ( fetch_attribute )
{
    ATTRIBUTE *attribute =
        attribute_fetch(
            attribute_name );
}

char *folder_attribute_prompt(
    char *attribute_name,
    int primary_key_index );

/* Usage */
FOLDER_ATTRIBUTE *folder_attribute_new(
    char *folder_name,
    char *attribute_name );

/* Process */
FOLDER_ATTRIBUTE *folder_attribute_calloc(
    void );

/* Usage */
boolean folder_attribute_exists(
    char *folder_name,
    char *attribute_name );

/* Process */
char *folder_attribute_primary_where_string(
    FOLDER_PRIMARY_KEY,
    ATTRIBUTE_PRIMARY_KEY,
    folder_name,
    attribute_name );

char *folder_attribute_exists_system_string(
    char *FOLDER_ATTRIBUTE_TABLE,
    char *folder_attribute_primary_where_string() );

boolean folder_attribute_exists_boolean(
    char *folder_attribute_exists_system_string() );

/* Usage */
FOLDER_ATTRIBUTE *folder_attribute_fetch(
    char *folder_name,
    char *attribute_name,
    boolean fetch_attribute );

/* Process */
char *folder_attribute_primary_where_string(
    FOLDER_PRIMARY_KEY,
    ATTRIBUTE_PRIMARY_KEY,
    folder_name,
    attribute_name );

char *folder_attribute_system_string(
    FOLDER_ATTRIBUTE_SELECT,
    FOLDER_ATTRIBUTE_TABLE,
    folder_attribute_primary_where_string() );

char *string_pipe_input(
    folder_attribute_system_string() );

FOLDER_ATTRIBUTE *folder_attribute_parse(
    string_pipe_input(),
    fetch_attribute );

/* Attributes */
char *folder_name;
char *attribute_name;
int primary_key_index;
int display_order;
boolean omit_insert;
boolean omit_insert_prompt;
boolean omit_update;
boolean additional_unique_index;
boolean additional_index;
boolean insert_required;
boolean lookup_required;
char *default_value;
char *prompt;
ATTRIBUTE *attribute;
```



FOLDER_ATTRIBUTE (2)

```
/* Usage */
LIST *folder_attribute_list(
    char *folder_name,
    LIST *role_attribute_exclude_lookup_name_list,
    boolean fetch_attribute,
    boolean cache_boolean );

/* Process */
if ( cache_boolean )
{
    LIST *folder_attribute_cache_list(
        folder_name,
        role_attribute_exclude_lookup_name_list,
        fetch_attribute );
}
else
{
    LIST *folder_attribute_fetch_list(
        folder_name,
        role_attribute_exclude_lookup_name_list,
        fetch_attribute );
}

/* Usage */
LIST *folder_attribute_cache_list(
    char *folder_name,
    LIST *role_attribute_exclude_lookup_name_list,
    boolean fetch_attribute );

/* Process */
static LIST *cache_list = {0};

if ( !cache_list )
{
    cache_list =
        folder_attribute_get_list(
            role_name,
            fetch_attribute );
}

LIST *folder_attribute_seek_list(
    folder_name,
    role_attribute_exclude_lookup_name_list,
    cache_list
        /* folder_attribute_list */ );

/* Usage */
LIST *folder_attribute_get_list(
    boolean fetch_attribute );

/* Process */
char *folder_attribute_system_string(
    FOLDER_ATTRIBUTE_SELECT,
    FOLDER_ATTRIBUTE_TABLE,
    (char *)0 /* where_string */ );

LIST *folder_attribute_system_list(
    folder_attribute_system_string(),
    fetch_attribute );
```

FOLDER_ATTRIBUTE (3)

```
/* Usage */
LIST *folder_attribute_system_list(
    char *folder_attribute_system_string(),
    boolean fetch_attribute );

/* Process */
FILE *appaserver_input_pipe(
    folder_attribute_system_string() );

LIST *list = list_new();

while ( string_input( input[], appaserver_input_pipe() );
{
    FOLDER_ATTRIBUTE *folder_attribute =
        folder_attribute_parse(
            input,
            fetch_attribute );

    list_set( list, folder_attribute );
}
pclose( appaserver_input_pipe() );

/* Usage */
char *folder_attribute_system_string(
    char *FOLDER_ATTRIBUTE_SELECT,
    char *FOLDER_ATTRIBUTE_TABLE,
    char *where_string );

/* Process */
/* Mysteriously, this sorts by primary_key_index */
/* then display_order, which is what we want. */
char *order_string =
    "folder,"
    "ifnull(display_order,0),"
    "ifnull(primary_key_index,0)";

snprintf(
    system_string[],
    sizeof ( system_string ),
    "select.sh \"%s\" %s \"%s\" \"%s\"",
    FOLDER_ATTRIBUTE_SELECT,
    FOLDER_ATTRIBUTE_TABLE,
    (where_string) ? where_string : "",
    order_string );

/* Usage */
char *folder_attribute_insert_statement(
    char *FOLDER_ATTRIBUTE_SELECT,
    char *FOLDER_ATTRIBUTE_TABLE,
    FOLDER_ATTRIBUTE *folder_attribute );

/* Process */
char *primary_string =
    attribute_insert_value_string(
        primary_key_index /* integer */,
        (char *)0 /* string */ );

char *display_order_string =
    attribute_insert_value_string(
        display_order /* integer */,
        (char *)0 /* string */ );

char *omit_insert_string =
    attribute_insert_value_string(
        0 /* integer */,
        (omit_insert) ? "y" : "" /* string */ );

char *omit_insert_prompt_string =
    attribute_insert_value_string(
        0 /* integer */,
        (omit_insert_prompt) ? "y" : ""  /* string */ );

char *omit_update_string =
    attribute_insert_value_string(
        0 /* integer */,
        (omit_update) ? "y" : "" /* string */ );

char *additional_unique_index_string =
    attribute_insert_value_string(
        0 /* integer */,
        (additional_unique_index) ? "y" : "" /* string */ );

char *additional_index_string =
    attribute_insert_value_string(
        0 /* integer */,
        (additional_index) ? "y" : "" /* string */ );

char *insert_required_string =
    attribute_insert_value_string(
        0 /* integer */,
        (insert_required) ? "y" : "" /* string */ );

char *lookup_required_string =
    attribute_insert_value_string(
        0 /* integer */,
        (lookup_required) ? "y" : "" /* string */ );

char *default_value_string =
    attribute_insert_value_string(
        0 /* integer */,
        folder_attribute->default_value /* string */ );

char *folder_attribute_insert_statement_string(
    char *FOLDER_ATTRIBUTE_SELECT,
    char *FOLDER_ATTRIBUTE_TABLE,
    char *folder_attribute->folder_name,
    char *folder_attribute->attribute_name,
    char *primary_string,
    char *display_order_string,
    char *omit_insert_string,
    char *omit_insert_prompt_string,
    char *omit_update_string,
    char *additional_unique_index_string,
    char *additional_index_string,
    char *insert_required_string,
    char *lookup_required_string,
    char *default_value_string );

/* Usage */
char *folder_attribute_list_string(
    char *folder_name );

/* Process */
LIST *folder_attribute_list(
    (char *)0 /* role_name */,
    folder_name,
    (LIST *)0 /* exclude_attribute_name_list */,
    0 /* not fetch_attribute */ );

LIST *folder_attribute_name_list(
    (char *)0 /* folder_name */,
    folder_attribute_list() );

return
char *list_delimited(
    folder_attribute_name_list(),
```

FOLDER_ATTRIBUTE (4)

```
/* Usage */
char *folder_attribute_where_string(
    LIST *folder_attribute_list,
    LIST *primary_key_list,
    LIST *primary_data_list );

/* Process */
static char where_string[];
char *ptr = where_string;

if (    !list_length( primary_key_list )
||      list_length( primary_key_list ) !=
        list_length( primary_data_list ) )
{
    exit( 1 );
}

list_rewind( primary_data_list );

for char *primary_key in primary_key_list
{
    if ( ptr != where_string )
    {
        ptr += sprintf( ptr, " and " );
    }

    FOLDER_ATTRIBUTE *folder_attribute =
        folder_attribute_seek(
            (char *)0 /* folder_name */,
            primary_key,
            folder_attribute_list );

    if ( folder_attribute )
    {
        char *datatype_name =
            folder_attribute->
                attribute->
                datatype_name;
    }

    char *primary_data =
        list_get(
            primary_data_list );

    if ( attribute_is_number( datatype_name ) )
    {
        ptr += sprintf( ptr,
            "%s = %s",
            primary_key,
            primary_data );
    }
    else
    {
        ptr += sprintf( ptr,
            "%s = '%s'",
            primary_key,
            primary_data );
    }
    list_next( data_list );
}

/* Usage */
char *folder_attribute_fetch_where_string(
    const char *FOLDER_PRIMARY_KEY,
    char *folder_name,
    LIST *role_attribute_exclude_lookup_name_list );

/* Process */
if ( list_length(
        role_attribute_exclude_lookup_name_list ) )
{
    char *folder_attribute_exclude_where_string(
        ATTRIBUTE_PRIMARY_KEY,
        role_attribute_exclude_lookup_name_list );

    sprintf( where_string[],
        "%s = '%s' and %s",
        FOLDER_PRIMARY_KEY,
        folder_name,
        folder_attribute_exclude_where_string() );
}
else
{
    sprintf( where_string[],
        FOLDER_PRIMARY_KEY,
        "%s = '%s'",
        folder_name );
}

/* Usage */
char *folder_attribute_primary_where_string(
    const char *FOLDER_PRIMARY_KEY,
    const char *ATTRIBUTE_PRIMARY_KEY,
    char *folder_name,
    char *attribute_name );

/* Process */
sprintf( static where_string[],
    "%s = '%s' and %s = '%s'",
    FOLDER_PRIMARY_KEY
    folder_name,
    ATTRIBUTE_PRIMARY_KEY,
    attribute_name );

/* Usage */
char *folder_attribute_exclude_where_string(
    const char *ATTRIBUTE_PRIMARY_KEY,
    LIST *role_attribute_exclude_lookup_name_list );

/* Process */
char *string_in_clause(
    role_attribute_exclude_lookup_name_list );

sprintf( static where_string[],
    "%s not in (%s)",
    ATTRIBUTE_PRIMARY_KEY,
    string_in_clause() );

free( string_in_clause() );
```

FOLDER_ATTRIBUTE (5)

```
/* Usage */
LIST *folder_attribute_append_isa_list(
    LIST *folder_attribute_list,
    LIST *relation_mto1_isa_list );

/* Process */
LIST *append_isa_list =
    list_copy(
        folder_attribute_list );

for RELATION_MTO1 *relation_mto1 in
    relation_mto1_isa_list
{
    LIST *folder_attribute_non_primary_list(
        relation_mto1->
            one_folder->
            folder_attribute_list );

    list_set_list(
        append_isa_list,
        folder_attribute_non_primary_list() );
}

/* Usage */
LIST *folder_attribute_seek_list(
    char *folder_name,
    LIST *role_attribute_exclude_lookup_name_list,
    LIST *folder_attribute_list );

/* Process */
boolean got_one = 0;
LIST *list = list_new();

for FOLDER_ATTRIBUTE *folder_attribute in
    folder_attribute_list
{
    if ( list_string_exists(
        folder_attribute->attribute_name,
        role_attribute_exclude_lookup_name_list ) )
    {
        continue;
    }
    if ( strcmp(
        folder_attribute->folder_name,
        folder_name ) == 0 )
    {
        list_set( list, folder_attribute );
        got_one = 1;
    }
    else
    {
        if ( got_one ) break;
    }
}

/* Usage */
FOLDER_ATTRIBUTE *folder_attribute_seek(
    char *folder_name,
    char *attribute_name,
    LIST *folder_attribute_list );

/* Process */
for FOLDER_ATTRIBUTE *folder_attribute in
    folder_attribute_list
{
    if ( folder_name )
    {
        if ( strcmp(
                folder_attribute->folder_name,
                folder_name ) != 0 )
        {
            continue;
        }
    }

    if ( strcmp(
            folder_attribute->attribute_name,
            attribute_name ) == 0 )
    {
        return folder_attribute;
    }
}
return NULL;

/* Usage */
LIST *folder_attribute_viewonly_attribute_name_list(
    LIST *folder_attribute_append_isa_list() );

/* Process */
for FOLDER_ATTRIBUTE *folder_attribute in
    folder_attribute_append_isa_list
{
    if ( folder_attribute->omit_update )
    {
        list_set(
            viewonly_attribute_name_list,
            folder_attribute->attribute_name );
    }
}

/* Usage */
int folder_attribute_date_name_list_length(
    LIST *folder_attribute_list );

/* Process */
for FOLDER_ATTRIBUTE *folder_attribute in
    folder_attribute_list
{
    boolean attribute_is_date(
        folder_attribute->attribute->datatype_name );

    boolean attribute_is_date_time(
        folder_attribute->attribute->datatype_name );

    boolean attribute_is_current_date_time(
        folder_attribute->attribute->datatype_name );

    if (    attribute_is_date()
    ||      attribute_is_date_time()
    ||      attribute_is_current_date_time() )
    {
        length++
    }
}
```

FOLDER_ATTRIBUTE (6)

```
/* Usage */
LIST *folder_attribute_name_list(
    char *folder_name,
    LIST *folder_attribute_[append_isa_]list );

/* Process */

/* Usage */
LIST *folder_attribute_folder_prefixed_name_list(
    LIST *folder_attribute_list );

/* Process */

/* Usage */
LIST *folder_attribute_primary_list(
    char *folder_name,
    LIST *folder_attribute_list );

/* Process */

/* Usage */
LIST *folder_attribute_non_primary_list(
    LIST *folder_attribute_list );

/* Process */

/* Usage */
LIST *folder_attribute_primary_key_list(
    LIST *folder_attribute_list );

/* Process */

/* Usage */
LIST *folder_attribute_non_primary_name_list(
    LIST *folder_attribute_list );

/* Process */

/* Usage */
LIST *folder_attribute_number_list(
    LIST *folder_attribute_list );


/* Process */

/* Usage */
LIST *folder_attribute_date_name_list(
    LIST *folder_attribute_list );

/* Process */

/* Usage */
LIST *folder_attribute_time_name_list(
    LIST *folder_attribute_list );

/* Process */

/* Usage */
LIST *folder_attribute_date_time_name_list(
    LIST *folder_attribute_list );

/* Process */

/* Usage */
LIST *folder_attribute_float_name_list(
    LIST *folder_attribute_list );

/* Process */

/* Usage */
LIST *folder_attribute_integer_name_list(
    LIST *folder_attribute_list );

/* Process */

/* Usage */
LIST *folder_attribute_number_name_list(
    LIST *folder_attribute_list );

/* Process */

/* Usage */
char *folder_attribute_sort_attribute_name(
    LIST *folder_attribute_list );


/* Process */

/* Usage */
char *folder_attribute_list_display(
    LIST *folder_attribute_list );

/* Process */

/* Usage */
LIST *folder_attribute_fetch_primary_key_list(
    char *folder_name );

/* Process */

/* Usage */
LIST *folder_attribute_name_list_attribute_list(
    LIST *folder_name_list );

/* Process */

/* Usage */
LIST *folder_attribute_lookup_required_name_list(
    LIST *folder_attribute_list );

/* Process */

/* Usage */
LIST *folder_attribute_insert_required_name_list(
    LIST *folder_attribute_list );

/* Process */

/* Usage */
LIST *folder_attribute_upload_filename_list(
    LIST *folder_attribute_list );

/* Process */

/* Usage */
char *folder_attribute_fetch_default_value(
    char *folder_name,
    char *attribute_name );

/* Process */
FOLDER_ATTRIBUTE *folder_attribute_fetch(
    folder_name,
    attribute_name,
    0 /* not fetch_attribute */ );

char *folder_attribute_fetch->default_value;
```

ATTRIBUTE (1)

```
/* Usage */
ATTRIBUTE *attribute_fetch(
    char *attribute_name );

/* Process */
static LIST *attribute_list = {0};

if ( !attribute_list )
{
    attribute_list = attribute_fetch_list();
}

ATTRIBUTE *attribute_seek(
    attribute_name,
    attribute_list );

/* Usage */
LIST *attribute_fetch_list( void );

/* Process */
char *appaserver_system_string(
    ATTRIBUTE_SELECT,
    ATTRIBUTE_TABLE,
    “1 = 1” /* where */ );

LIST *attribute_system_list(
    appaserver_system_string() );

/* Usage */
LIST *attribute_system_list(
    char *appaserver_system_string() );

/* Process */
LIST *list = list_new();

FILE *appaserver_input_pipe(
    appaserver_system_string );

for char input[] in
    string_input(
        appaserver_input_pipe ) )
{
    list_set(
        list,
        attribute_parse(
            input ) );
}
pclose( appaserver_input_pipe() );

/* Usage */
ATTRIBUTE *attribute_parse(
    char *input );

/* Process */
ATTRIBUTE *attribute_new(
    strdup( attribute_name ) );

/* Usage */
ATTRIBUTE *attribute_new(
    char *attribute_name );

/* Process */
ATTRIBUTE *attribute_calloc(
    void );

/* Attributes */
char *attribute_name;
char *datatype_name;
int width;
int float_decimal_places;
char *hint_message;
```

ATTRIBUTE (2)

```
/* Usage */
ATTRIBUTE *attribute_seek(
    char *attribute_name,
    LIST *attribute_list );

/* Process */
for ATTRIBUTE *attribute in attribute_list
{
    if ( strcmp(
        attribute->attribute_name,
        attribute_name ) == 0 )
    {
        return attribute;
    }
}
return NULL;

/* Usage */
char *attribute_insert_statement(
    char *ATTRIBUTE_SELECT,
    char *ATTRIBUTE_TABLE,
    ATTRIBUTE *attribute );

/* Process */
char *width_string =
    attribute_insert_value_string(
        width /* integer */,
        (char *)0 /* string */ );

char *float_string =
    attribute_insert_value_string(
        float_decimal_places /* integer */,
        (char *)0 /* string */ );

char *hint_string =
    attribute_insert_value_string(
        0 /* integer */,
        hint_message /* string */ );

char *attribute_insert_statement_string(
    char *ATTRIBUTE_SELECT,
    char *ATTRIBUTE_TABLE,
    char *attribute->attribute_name,
    char *attribute->datatype_name,
    char *width_string,
    char *float_string,
    char *hint_string );

free( width_string );
free( float_string );
free( hint_string );

/* Usage */
DICTIONARY *attribute_query_dictionary(
    const char *QUERY_RELATION_OPERATOR_PREFIX,
    const char *QUERY_EQUAL,
    LIST *attribute_name_list,
    LIST *attribute_data_list );

/* Process */
DICTIONARY *query_dictionary = dictionary_small();

list_rewind( attribute_data_list );

for char *attribute_name in attribute_name_list
{
    char *attribute_datum =
        list_get(
            attribute_data_list );

    char *security_sql_injection_escape(
        SECURITY_ESCAPE_CHARACTER_STRING,
        attribute_datum );

    void dictionary_set(
        query_dictionary,
        attribute_name,
        security_sql_injection_escape() );

    char *query_relation_key(
        QUERY_RELATION_OPERATOR_PREFIX,
        attribute_name );

    void dictionary_set(
        query_dictionary,
        strdup(
            query_relation_key() );
        QUERY_EQUAL );
    }
    list_next( attribute_data_list );
}

/* Usage */
char *attribute_trim_number_characters(
    char *number );

/* Process */
string_trim_character_array(
    number,
    ",$" );

/* Usage */
char *attribute_insert_value_string(
    int integer,
    char *string );

/* Process */
if ( !integer && ( !string || !*string ) )
{
    strcpy( value_string[], "" );
}
else
if ( integer )
{
    sprintf( value_string[],
    "%d",
    integer );
}
else
{
    sprintf( value_string[],
    "'%s'",
    string );
}
```

ATTRIBUTE (3)

```
/* Usage */
void attribute_folder_name_list_stdout(
    char *folder_name );

/* Process */
LIST *folder_attribute_list(
    folder_name,
    (LIST *)0 /* exclude_attribute... */,
    0 /* not fetch_attribute */,
    0 /* not cache_boolean */ );

LIST *folder_attribute_name_list(
    folder_attribute_list );

void attribute_name_list_sort_stdout(
    folder_attribute_name_list() );

/* Usage */
void attribute_name_list_stdout(
    const char *ATTRIBUTE_TABLE,
    const char *ATTRIBUTE_PRIMARY_KEY );

/* Process */
snprintf(
    system_string[],
    sizeof ( system_string ),
    "select.sh %s %s '' select",
    ATTRIBUTE_PRIMARY_KEY,
    ATTRIBUTE_TABLE );

system( system_string );

/* Usage */
void attribute_name_list_sort_stdout(
    LIST *attribute_name_list );

/* Process */
FILE *appaserver_output_pipe( "sort -u" );

for char *attribute_name in attribute_name_list
{
    fprintf( appaserver_output_pipe(),
        "%s\n",
        attribute_name );
}
pclose( appaserver_output_pipe );

/* Usage */
char *attribute_name_trim_datatype(
    char *attribute_name,
    boolean attribute_is_date(),
    boolean attribute_is_time() );

/* Process */
if ( attribute_is_date )
{
    if ( strcmp(
            right_string( attribute_name, 5 ),
            "_date" ) == 0 )
    {
        strcpy( attribute_trimmed[], attribute_name );
        string_trim_right( attribute_trimmed, 5 );
    }
}
else
if ( attribute_is_time )
{
    if ( strcmp(
            right_string( attribute_name, 5 ),
            "_time" ) == 0 )
    {
        strcpy( attribute_trimmed[], attribute_name );
        string_trim_right( attribute_trimmed, 5 );
    }
}

/* Usage */
LIST *attribute_list(
    void );

/* Process */
static LIST *list = {0};

if ( list ) return list;

char *attribute_system_string(
    ATTRIBUTE_SELECT,
    ATTRIBUTE_TABLE,
    (char *)0 /* where */ );

LIST *list =
    attribute_system_list(
        attribute_system_string() );

/* Usage */
boolean attribute_exists(
    char *attribute_name,
    LIST *attribute_list );

/* Process */
attribute_seek(
    attribute_name,
    attribute_list );

if ( attribute_seek() )
    return 1;
else
    return 0;
```

ATTRIBUTE (4)

```
/* Public */
LIST *attribute_extract_float_list(
    LIST *attribute_list );

char *attribute_full_attribute_name(
    char *appaserver_table_name(),
    char *attribute_name );

boolean attribute_is_date(
    char *datatype_name );

boolean attribute_is_time(
    char *datatype_name );

boolean attribute_is_date_time(
    char *datatype_name );

boolean attribute_is_current_date_time(
    char *datatype_name );

boolean attribute_is_current_date(
    char *datatype_name );

boolean attribute_is_float(
    char *datatype_name );

boolean attribute_is_integer(
    const char *ATTRIBUTE_DATATYPE_INTEGER,
    char *datatype_name );

boolean attribute_is_number(
    char *datatype_name );

boolean attribute_is_character(
    const char *ATTRIBUTE_DATATYPE_CHARACTER,
    char *datatype_name );

boolean attribute_is_notepad(
    const char *ATTRIBUTE_DATATYPE_NOTEPAD,
    char *datatype_name );

boolean attribute_is_password(
    char *datatype_name );

boolean attribute_is_encrypt(
    char *datatype_name );

boolean attribute_is_upload(
    char *datatype_name );

boolean attribute_is_timestamp(
    char *datatype_name );

boolean attribute_is_yes_no(
    char *attribute_name );

int attribute_width(
    char *attribute_name );

char *attribute_database_datatype(
    char *datatype_name,
    int width,
    int float_decimal_places,
    int primary_key_index,
    char *default_value );

char *attribute_delete_statement(
    const char *ATTRIBUTE_TABLE,
    char *attribute_name );

LIST *attribute_data_list(
    char delimiter,
    char *data_list_string );
```

FOREIGN_ATTRIBUTE (1)

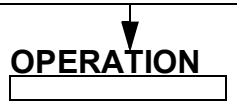
```
/* Usage */
LIST *foreign_attribute_list(
    char *many_folder_name,
    char *one_folder_name,
    char *related_attribute_name );

/* Process */
static LIST *cache_list = {0};

if ( !cache_list )
    cache_list =
        foreign_attribute_fetch_list();

LIST *foreign_attribute_seek_list(
    many_folder_name,
    one_folder_name,
    related_attribute_name,
    cache_list /* foreign_attribute_list */ );

/* Usage */
FOREIGN_ATTRIBUTE *foreign_attribute_parse(
    char *input );

/* Process */
FOREIGN_ATTRIBUTE *foreign_attribute_new(
    strdup( folder_name ),
    strdup( related_folder_name ),
    strdup( related_attribute_name ),
    strdup( foreign_attribute_name ) );

/* Usage */
FOREIGN_ATTRIBUTE *foreign_attribute_new(
    char *folder_name,
    char *related_folder_name,
    char *related_attribute_name,
    char *foreign_attribute_name );

/* Process */
FOREIGN_ATTRIBUTE *foreign_attribute_calloc(
    void );

/* Usage */
LIST *foreign_attribute_name_list(
    LIST *foreign_attribute_list() );

/* Process */
LIST *name_list = list_new();

for FOREIGN_ATTRIBUTE *foreign_attribute in
    foreign_attribute_list
{
    list_set(
        name_list,
        foreign_attribute->foreign_attribute_name );
}

/* Usage */
LIST *foreign_attribute_fetch_list(
    void );

/* Process */
char *foreign_attribute_system_string(
    FOREIGN_ATTRIBUTE_SELECT,
    FOREIGN_ATTRIBUTE_TABLE );

LIST *foreign_attribute_system_list(
    foreign_attribute_system_string();

/* Usage */
LIST *foreign_attribute_system_list(
    char *foreign_attribute_system_string() );

/* Process */
char *appaserver_input_pipe(
    foreign_attribute_system_string );

while ( string_input( input[], appaserver_input_pipe() )
{
    if ( !list ) list = list_new();

    list_set(
        list,
        foreign_attribute_parse(
            input ) );
}
pclose( appaserver_input_pipe() );

/* Usage */
LIST *foreign_attribute_seek_list(
    char *many_folder_name,
    char *one_folder_name,
    char *related_attribute_name,
    LIST *foreign_attribute_list() );

/* Process */
LIST *list = {0};

for FOREIGN_ATTRIBUTE *foreign_attribute in
    foreign_attribute_list
{
    if (strcmp(
            foreign_attribute->folder_name,
            many_folder_name ) == 0
    &&strcmp(
            foreign_attribute->related_folder_name,
            one_folder_name ) == 0
    &&strcmp(
            foreign_attribute->related_attribute_name,
            related_attribute_name ) == 0 )
    {
        if ( !list ) list = list_new();

        list_set(
            list,
            foreign_attribute );
    }
}
```

FOREIGN_ATTRIBUTE (2)

```
/* Usage */
char *foreign_attribute_system_string(
    FOREIGN_ATTRIBUTE_SELECT,
    FOREIGN_ATTRIBUTE_TABLE );

/* Process */
sprintf( system_string[],
    "select.sh '%s' %s '1 = 1' folder,foreign_key_index",
    FOREIGN_ATTRIBUTE_SELECT,
    FOREIGN_ATTRIBUTE_TABLE );

/* Usage */
char *foreign_attribute_translate(
    char *related_attribute_name
        /* primary_key */,
    LIST *foreign_attribute_list();

/* Process */
FOREIGN_ATTRIBUTE *foreign_attribute_related_seek(
    char *related_attribute_name,
    LIST *foreign_attribute_list );

FOREIGN_ATTRIBUTE *foreign_attribute =
    list_index_seek(
        foreign_attribute_related_seek()->
            foreign_key_index,
            foreign_attribute_list );

return
foreign_attribute->foreign_attribute_name;

/* Attributes */
char *folder_name;
char *related_folder_name;
char *related_attribute_name;
char *foreign_attribute_name;
int foreign_key_index;
```

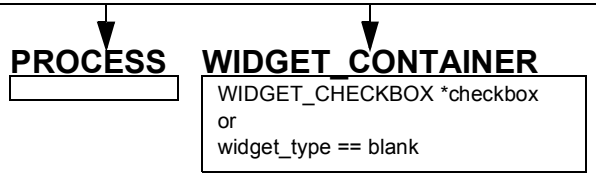
FOLDER_OPERATION

```
/* Usage */
LIST *folder_operation_list(
    char *folder_name,
    char *role_name,
    boolean fetch_operation,
    boolean fetch_process );

/* Process */
char *folder_operation_where(
    const char *FOLDER_PRIMARY_KEY,
    char *folder_name,
    char *role_name );

char *folder_operation_system_string(
    char *FOLDER_OPERATION_SELECT,
    char *FOLDER_OPERATION_TABLE,
    char *folder_operation_where() );

LIST *folder_operation_system_list(
    folder_operation_system_string(),
    fetch_operation,
    fetch_process );

/* Usage */
LIST *folder_operation_system_list(
    char *folder_operation_system_string(),
    boolean fetch_operation,
    boolean fetch_process );

/* Process */
LIST *list = list_new();

FILE *appaserver_input_pipe(
    folder_operation_system_string );

while ( string_input( input[], appaserver_input_pipe() )
{
    list_set(
        list,
        FOLDER_OPERATION *folder_operation_parse(
            input,
            fetch_operation,
            fetch_process ) );
}
void pclose( appaserver_input_pipe() );

/* Usage */
FOLDER_OPERATION *folder_operation_parse(
    char *input,
    boolean fetch_operation,
    boolean fetch_process );

/* Process */
FOLDER_OPERATION *folder_operation_new(
    strdup( folder_name ),
    strdup( role_name ),
    strdup( operation_name ) );

/* Usage */
FOLDER_OPERATION *folder_operation_new(
    char *folder_name,
    char *role_name,
    char *operation_name );

/* Process */
FOLDER_OPERATION *folder_operation_calloc(
    void );

if ( fetch_operation )
{
    OPERATION *operation =
        operation_fetch(
            operation_name,
            fetch_process );
}

/* Public */
LIST *folder_operation_name_list(
    LIST *folder_operation_list() );

boolean folder_operation_delete_boolean(
    LIST *folder_operation_list() );

/* Attributes */
char *folder_name;
char *role_name;
char *operation_name;
OPERATION *operation;
```



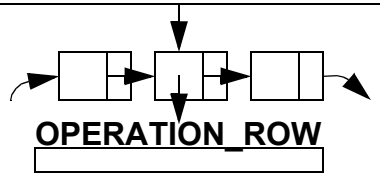
OPERATION

```
/* Usage */
OPERATION *operation_fetch(
   char *operation_name,
   boolean fetch_process );

/* Process */
char *operation_primary_where(
   char *operation_name );

char *operation_system_string(
   char *OPERATION_SELECT,
   char *OPERATION_TABLE,
   char *operation_primary_where );

OPERATION *operation_parse(
   string_pipe_input(
      operation_system_string() ),
   fetch_process );

/* Usage */
OPERATION *operation_parse(
   char *input,
   boolean fetch_process );

/* Process */
OPERATION *operation_calloc(
   void );

boolean operation_output_boolean(
   char *output_yn );

boolean operation_delete_boolean(
   char *operation_name );

boolean operation_drilldown_boolean(
   char *operation_name );

char *operation_image_source(
   boolean operation_delete_boolean(),
   boolean operation_drilldown_boolean() );

char *operation_delete_warning_javascript(
   const char *
      OPERATION_DELETE_WARNING_JAVASCRIPT,
   boolean operation_delete_boolean() );

WIDGET_CONTAINER *operation_widget_container(
   char *operation_name,
   char *operation_image_source(),
   char *operation_delete_warning_javascript() );

if ( fetch_process )
{
   char *appaserver_parameter_mount_point();

   PROCESS *process =
      process_fetch(
         operation_name,
         (char *)0 /* document_root_directory */,
         (char *)0 /* relative_source_directory */,
         1 /* check_executable_inside_filesystem */,
         appaserver_parameter_mount_point() );
}

/* Usage */
char *operation_html(
   WIDGET_CONTAINER *widget_container,
   char *state,
   int row_number,
   char *background_color,
   boolean delete_mask_boolean );

/* Process */
if ( delete_mask_boolean )
{
   char prompt_display[];

   string_initial_capital(
      prompt_display,
      widget_container->checkbox->prompt );

   return
      strdup( prompt_display );
}

char *widget_container_html(
   state,
   row_number /* -1 will omit index suffix */,
   (LIST *) /* query_row_cell_list */,
   (RELATION_JOIN_ROW *)0 /* relation_join_row */,
   background_color,
   widget_container );

/* Usage */
int operation_row_checked_count(
   DICTIONARY *dictionary_separate->
      operation_dictionary,
   char *operation_name,
   int highest_index );

/* Process */

/* Usage *
OPERATION *operation_blank_new(
   void );

/* Process */
OPERATION *operation_calloc();

operation_calloc()->widget_container =
   widget_container_new(
      blank /* widget_type */,
      (char *)0 /* widget_name */ );

/* Attributes */
char *operation_name;
boolean output_boolean;
PROCESS *process;
boolean delete_boolean;
boolean drilldown_boolean;
char *image_source;
char *delete_warning_javascript;
WIDGET_CONTAINER *widget_container;
```



OPERATION_ROW_LIST

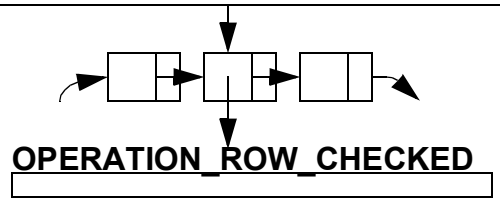
```
/* Usage */
OPERATION_ROW_LIST *operation_row_list_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *target_frame,
    char *state,
    LIST *folder_operation_list(),
    LIST *folder->folder_attribute_primary_key_list,
    LIST *folder->folder_attribute_name_list,
    DICTIONARY *dictionary_separate->
        operation_dictionary,
    DICTIONARY *dictionary_separate->
        no_display_dictionary,
    DICTIONARY *dictionary_separate->
        multi_row_dictionary,
    DICTIONARY *dictionary_separate->
        drillthru_dictionary,
    char *update_results_string,
    char *update_error_string );

/* Process */
OPERATION_ROW_LIST *operation_row_list_calloc(
    void );

/* Returns >=  0 */
int dictionary_highest_index(
    operation_dictionary );

if ( !dictionary_highest_index() ) return NULL;

DICTIONARY *operation_row_list_dictionary(
    char *DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
    char *DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
    DICTIONARY *operation_dictionary,
    DICTIONARY *no_display_dictionary,
    DICTIONARY *drillthru_dictionary );

LIST *list = list_new();

for (  row_number = 1;
       row_number <=
           dictionary_highest_index();
       row_number++ )
{
    DICTIONARY *dictionary_single_row(
        folder_attribute_name_list,
        row_number,
        multi_row_dictionary );

    if ( !dictionary_length( dictionary_single_row() ) )
            continue;

    list_set(
        list,
        operation_row_new(
            application_name,
            session_key,
            login_name,
            role_name,
            folder_name,
            target_frame,
            state,
            folder_operation_list,
            folder_attribute_primary_key_list,
            update_results_string,
            update_error_string,
            dictionary_highest_index(),
            operation_row_list_dictionary(),
            row_number,
            dictionary_single_row() ) );
}

boolean operation_row_list_output_boolean(
    LIST *this->list /* operation_row_list */ );

/* Usage */
/* Returns error_message_list_string or null */
char *operation_row_list_execute(
    OPERATION_ROW_LIST *operation_row_list );

/* Process */
char operation_error_message_list_string[];
char *ptr = operation_error_message_list_string;

for OPERATION_ROW *operation_row
    in operation_row_list->list
{
    if ( !list_rewind(
            operation_row->
                operation_row_checked_list ) )
    {
            continue;
    }

    for OPERATION_ROW_CHECKED *
        operation_row_checked in
            operation_row->operation_row_checked_list )
    {
        char *operation_error_message =
            char *operation_row_checked_execute(
                operation_row_checked->command_line,
                operation_row_list->output_boolean );

        if ( operation_error_message )
        {
            if ( ptr != operation_error_message_list_string )
            {
                ptr += sprintf( ptr, "\n<br>" );

                ptr += sprintf( ptr,
                    "%s",
                    operation_error_message );
            }
        }
    }

/* Attributes */
int dictionary_highest_index;
DICTIONARY *dictionary;
LIST *list;
boolean output_boolean;
```



OPERATION_ROW

```
/* Usage */
OPERATION_ROW *operation_row_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *target_frame,
    char *state,
    LIST *folder_operation_list(),
    LIST *folder_attribute_primary_key_list,
    char *update_results_string,
    char *update_error_string,
    int dictionary_highest_index(),
    DICTIONARY *operation_row_list_dictionary(),
    int row_number,
    DICTIONARY *dictionary_single_row() );

/* Process */
OPERATION_ROW *operation_row_calloc(
    void );
```

```
LIST *operation_row_checked_list = list_new();

char *appaserver_error_filename(
    application_name );

for FOLDER_OPERATION *folder_operation in
    folder_operation_list
{
    if ( operation_row_checked(
            DICTIONARY *operation_dictionary,
            int row_number,
            char *folder_operation->operation_name ) )
    {
        list_set(
            operation_row_checked_list,
            operation_row_checked_new(
                session_key,
                login_name,
                role_name,
                folder_name,
                target_frame,
                state,
                folder_attribute_primary_key_list,
                post_table_edit_update_results_string,
                dictionary_highest_index,
                operation_row_list_dictionary,
                row_number,
                dictionary_single_row,
                folder_operation->operation_name,
                folder_operation->
                    operation->
                    process->
                    command_line,
                folder_operation->
                    operation->
                    output_boolean,
                appaserver_error_filename() ) );
    }
}

/* Attributes */
LIST *operation_row_checked_list;
char *appaserver_error_filename;
```



OPERATION_ROW_CHECKED (1)

```
/* Usage */
OPERATION_ROW_CHECKED *
    operation_row_checked_new(
        char *session_key,
        char *login_name,
        char *role_name,
        char *folder_name,
        char *target_frame,
        char *state,
        LIST *folder_attribute_primary_key_list,
        char *update_results_string,
        char *update_error_string,
        int dictionary_highest_index,
        DICTIONARY *operation_row_list_dictionary,
        int row_number,
        DICTIONARY *dictionary_single_row,
        char *operation_name,
        char *process_command_line,
        boolean output_boolean,
        char *appaserver_error_filename() );

/* Process */
boolean operation_row_checked_boolean(
    DICTIONARY *operation_row_list_dictionary,
    int row_number,
    char *operation_name );

if ( !operation_row_checked_boolean() ) return NULL;

OPERATION_ROW_CHECKED *
    operation_row_checked_calloc(
        void );

int operation_row_checked_count(
    int dictionary_highest_index,
    DICTIONARY *operation_row_list_dictionary,
    char *operation_name );

LIST *primary_key_data_list =
    dictionary_data_list(
        LIST *folder_attribute_primary_key_list,
        DICTIONARY *dictionary_single_row );

char *operation_row_checked_command_line(
    ATTRIBUTE_MULTI_KEY_DELIMITER,
    session_key,
    login_name,
    role_name,
    folder_name,
    target_frame,
    state,
    update_results_string,
    update_error_string,
    operation_row_list_dictionary,
    row_number,
    dictionary_single_row,
    operation_name,
    process_command_line,
    appaserver_error_filename,
    operation_row_checked_count,
    getpid() /* parent_process_id */,
    primary_key_data_list );

/* Usage */
boolean operation_row_checked_boolean(
    DICTIONARY *operation_row_list_dictionary,
    int row_number,
    char *operation_name );

/* Process */
char *operation_row_checked_key(
    row_number,
    operation_name );

boolean dictionary_key_exists(
    operation_row_list_dictionary,
    operation_row_checked_key() );

/* Usage */
int operation_row_checked_count(
    int dictionary_highest_index,
    DICTIONARY *operation_row_list_dictionary,
    char *operation_name );

/* Process */
int count = 0;

for (  int row_number = 1;
       row_number <= dictionary_highest_index;
       row_number++ )
{
    boolean operation_row_checked_boolean(
        operation_row_list_dictionary,
        row_number,
        operation_name );

    if ( operation_row_checked_boolean() ) count++;
}

/* Usage */
char *operation_row_checked_key(
    int row_number,
    char *operation_name );

/* Process */
sprintf( static key[],
    "%s_%d",
    operation_name,
    row_number );

/* Attributes */
boolean output_boolean;
int count;
char *row_number_string;
LIST *primary_key_data_list;
char *command_line;
```

OPERATION_ROW_CHECKED (2)

```
/* Usage */
/* Returns operation_error_message or null */
char *operation_row_checked_execute(
    char *operation_row_checked->command_line,
    boolean operation_row_list_output_boolean() );

/* Process */
if ( operation_row_list_output_boolean )
{
    security_system(
        SECURITY_FORK_CHARACTER,
        SECURITY_FORK_STRING,
        command_line );
}
else
{
    string_pipe_fetch( command_line );
}

/* Usage */
char *operation_row_checked_command_line(
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *target_frame,
    char *state,
    char *update_results_string,
    char *update_error_string,
    DICTIONARY *operation_row_list_dictionary,
    int row_number,
    DICTIONARY *dictionary_single_row,
    char *operation_name,
    char *process_command_line,
    char *appaserver_error_filespecification,
    int operation_row_checked_count,
    pid_t parent_process_id,
    LIST *primary_key_data_list );

/* Process */
char *process_replace_command_line(
    UPDATE_PREUPDATE_PREFIX,
    (char *)0 /* application_name */,
    session_key,
    login_name,
    role_name,
    folder_name,
    target_frame,
    state,
    operation_name /* process_name */,
    (char *)0 /* many_folder_name */,
    (char *)0 /* one_folder_name */,
    (char *)0 /* related_column */,
    update_results_string,
    update_error_string,
    operation_row_list_dictionary,
    dictionary_single_row,
    (char *)0 /* where_string */,
    row_number,
    operation_row_checked_count /* row_count */,
    parent_process_id,
    primary_key_data_list,
    "y" /* execute_yn */,
    (LIST *)0 /* insert_datum_list */,
    (LIST *)0 /* update_attribute_list */,
    (LIST *)0 /* query_cell_list */,
    appaserver_error_filespecification,
    process_command_line /* input_command_line */ );
```

OPERATION_SEMAPHORE (to retire; use PROCESS_ROW_NUMBER_PLACEHOLDER)

```
/* Usage */
OPERATION_SEMAPHORE *
    operation_semaphore_new(
        char *application_name,
        char *operation_name,
        char *appaserver_parameter->data_directory,
        pid_t parent_process_id,
        int operation_row_checked_count() );

/* Process */
OPERATION_SEMAPHORE *
    operation_semaphore_calloc(
        void );

char *operation_semaphore_filename(
    char *application_name,
    char *operation_name,
    char *data_directory,
    pid_t parent_process_id );

boolean operation_semaphore_group_first_time(
    operation_semaphore_filename() );

if ( operation_semaphore_group_first_time() )
{
    int row_current = 1;
}
else
{
    int row_current =
        operation_semaphore_row_current(
            operation_semaphore_filename() );
    int row_current++
}

boolean operation_semaphore_group_last_time(
    int row_current,
    int operation_row_checked_count );

if ( operation_semaphore_group_last_time() )
{
    if ( !operation_semaphore_group_first_time() )
    {
        void operation_semaphore_remove_file(
            char *operation_semaphore_filename() );
    }
}
else
{
    void operation_semaphore_write(
        operation_semaphore_filename(),
        row_current );
}

/* Usage */
boolean operation_semaphore_group_first_time(
    char *operation_semaphore_filename() );

/* Process */
boolean !application_file_exists_boolean(
    operation_semaphore_filename );

/* Usage */
int operation_semaphore_row_current(
    char *operation_semaphore_filename() );

/* Process */
FILE *appaserver_input_file(
    operation_semaphore_filename );

string_input( buffer[], appaserver_input_file() );
fclose( appaserver_input_file() );
int atoi( buffer );

/* Usage */
void operation_semaphore_write(
    char *operation_semaphore_filename(),
    int row_current );

/* Process */
FILE *appaserver_output_file(
    operation_semaphore_filename );

fprintf(
    appaserver_output_file(),
    "%d\n",
    row_current );

fclose( appaserver_output_file() );

/* Attributes */
char *filename;
boolean group_first_time;
int row_current;
boolean group_last_time;
```

SEMAPHORE

```
/* Usage */
SEMAPHORE *semaphore_new(
    key_t key );

/* Process */
SEMAPHORE *semaphore_calloc(
    void );

int semaphore_id( key );

/* Usage */
int semaphore_id(
    key_t key );

/* Process */
int id = semget( key, 1, 0600|IPC_CREAT|IPC_EXCL );

if ( id > -1 )
{
    int semctl( id, 0, SETVAL, 1 );
    if ( semctl() == -1 ) exit( 1 );
}
else
{
    id = semget( key, 1, 0 );
}

/* Usage */
void semaphore_wait(
    int semaphore->id );

/* Process */
struct sembuf semaphore_buffer;

semaphore_buffer.sem_num = 0;
semaphore_buffer.sem_op = -1;
semaphore_buffer.sem_flg = SEM_UNDO;

if ( semop( id, &semaphore_buffer, 1 ) == -1 )
    exit( 1 );

/* Usage */
void semaphore_signal(
    int semaphore->id );

/* Process */
struct sembuf semaphore_buffer;

semaphore_buffer.sem_num = 0;
semaphore_buffer.sem_op = 1;
semaphore_buffer.sem_flg = SEM_UNDO;

if ( semop( id, &semaphore_buffer, 1 ) == -1 )
        exit( 1 );

/* Attributes */
key_t key;
int id;
```

QUERY (1)

```
/* Usage */
char *query_dictionary_extract(
    DICTIONARY *dictionary,
    char *attribute_name,
    /* Mutually exclusive */
    char *PROMPT_LOOKUP_FROM_PREFIX,
    /* Mutually exclusive */
    char *PROMPT_LOOKUP_TO_PREFIX,
    boolean attribute_is_number() );

/* Process */
char *dictionary_get(
    attribute_name,
    dictionary );

if ( dictionary_get() )
{
    if ( attribute_is_number )
    {
        attribute_trim_number_characters(
            dictionary_get() );
    }
    return dictionary_get();
}

char *query_extract_key(
    char *attribute_name,
    char *PROMPT_LOOKUP_FROM_PREFIX,
    char *PROMPT_LOOKUP_TO_PREFIX );

char *dictionary_get(
    query_extract_key(),
    dictionary );

if ( dictionary_get() )
{
    if ( attribute_is_number )
    {
        attribute_trim_number_characters(
            dictionary_get() );
    }
}
return dictionary_get();

/* Usage */
char *query_table_name(
    char *folder_name,
    int relation_mto1_isa_list_length,
    char *attribute_not_null );

/* Process */
if (    !relation_mto1_isa_list_length
&&  !attribute_not_null )
{
    return (char *)0;
}

char *strdup(
    appaserver_table_name(
        folder_name ) );

/* Usage */
char *query_order_string(
    LIST *folder_attribute_primary_key_list,
    LIST *folder_attribute_name_list,
    DICTONARY *sort_dictionary );

/* Process */
int dictionary_length( sort_dictionary );

if (    dictionary_length()
&&  folder_attribute_name_list )
{
    LIST *dictionary_key_list( sort_dictionary );

    char *query_key_list_order_string(
        ATTRIBUTE_MULTI_KEY_DELIMITER,
        FORM_SORT_ASCEND_LABEL,
        FORM_SORT_DESCEND_LABEL,
        folder_attribute_name_list,
        dictionary_key_list()
            /* sort_dictionary_key_list */ );
}

if ( !query_key_list_order_string() )
{
    char *list_delimited_string(
        folder_attribute_primary_key_list,
        ',' /* delimiter */ );
}

/* Usage */
char *query_key_list_order_string(
    const char ATTRIBUTE_MULTI_KEY_DELIMITER,
    const char *FORM_SORT_ASCEND_LABEL,
    const char *FORM_SORT_DESCEND_LABEL,
    LIST *folder_attribute_name_list,
    LIST *sort_dictionary_key_list );

/* Process */
char *first_key = list_first( sort_dictionary_key_list );

int string_strncmp(
    first_key,
    FORM_SORT_ASCEND_LABEL );

if ( string_strncmp() == 0 )
{
    char *attribute_name_list_string =
        first_key + strlen( FORM_SORT_ASCEND_LABEL );

    char *query_key_order_string(
        ATTRIBUTE_MULTI_KEY_DELIMITER,
        folder_attribute_name_list,
        attribute_name_list_string,
        0 /* not descend_boolean */ );
}

int string_strncmp(
    first_key,
    FORM_SORT_DESCEND_LABEL );

if ( string_strncmp() == 0 )
{
    char *attribute_name_list_string =
        first_key + strlen( FORM_SORT_DESCEND_LABEL );

    char *query_key_order_string(
        ATTRIBUTE_MULTI_KEY_DELIMITER,
        folder_attribute_name_list,
        attribute_name_list_string,
        1 /* descend_boolean */ );
}
```

QUERY (2)

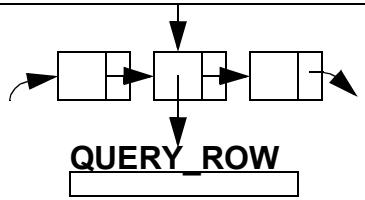
```
/* Usage */
char *query_key_order_string(
    const char ATTRIBUTE_MULTI_KEY_DELIMITER,
    LIST *folder_attribute_name_list,
    char *attribute_name_list_string,
    boolean descend_boolean );

/* Process */
char order_string[];
char *ptr = order_string;

for (  int p = 0;
       piece(
          attribute_name[],
          ATTRIBUTE_MULTI_KEY_DELIMITER,
          attribute_name_list_string,
          p );
       p++ )
{
    boolean list_string_boolean(
       attribute_name,
       folder_attribute_name_list );

    if ( list_string_boolean() )
    {
       if ( ptr != order_string ) ptr += sprintf( ptr, “,” );

       ptr += sprintf( ptr,
          “%s”,
          attribute_name );

       if ( descend_boolean )
            ptr += sprintf( ptr, “ desc” );
    }
}

/* Usage */
DICTIONARY *query_fetch_dictionary(
    LIST *select_name_list,
    char *appaserver_table_name(),
    LIST *primary_key_list,
    LIST *primary_data_list );

/* Process */
LIST *query_drop_down_datum_list = list_new();

char *appaserver_select(
    select_name_list
       /* folder_attribute_name_list */ );

list_rewind( primary_data_list );

for char *primary_key in primary_key_list
{
    QUERY_DROP_DOWN_DATUM *
       query_drop_down_datum =
          query_drop_down_data_new(
             (char *)0 /* many_table_name */,
             primary_key /* foreign_key */,
             list_get( primary_data_list )
                /* datum */ );

    list_set(
       query_drop_down_datum_list,
       query_drop_down_datum );

    list_next( primary_data_list );
}

char *query_drop_down_datum_list_where(
    query_drop_down_datum_list,
    “and” /* conjunction */ );

char *appaserver_system_string(
    appaserver_select(),
    appaserver_table_name,
    query_drop_down_datum_list_where() );

DICTIONARY *dictionary_name_list_fetch(
    appaserver_system_string(),
    select_name_list,
    SQL_DELIMITER );

/* Usage */
char *query_drop_down_key(
    LIST *foreign_key_list,
    int index,
    char MULTI_ATTRIBUTE_KEY_DELIMITER );

/* Process */

/* Usage */
char *query_system_string(
    char *application_name,
    char *select_string,
    char *from_string,
    char *where_string,
    char *group_by_string,
    char *order_string,
    int max_rows );

/* Process */
if ( !where_string || !*where_string ) where_string = “1 = 1”;

snprintf(
    system_string[],
    sizeof ( system_string ),
    “select.e  application=%s”
    “              select=\”%s\””
    “              folder=%s”
    “              where=\”%s\””
    “              group=\”%s\”
    “              order=\”%s\””
    “              maxrows=%d”
    “              quick=%s”,
    application_name,
    select_string,
    from_string,
    where_string,
    (group_by_string) ? group_by_string : “”,
    (order_string) ? order_string : “”,
    max_rows,
    QUERY_QUICK_YES_NO );
```

QUERY (3)

```
/* Usage */
char *query_from_string(
    char *folder_name,
    LIST *mto1_isa_related_folder_list,
    char *row_security_role_update_list_from );

/* Process */
char from_string[];
char *ptr = from_string;

char *appaserver_table_name( folder_name );

ptr += sprintf( ptr, "%s", appaserver_table_name() );

if ( list_length( mto1_isa_related_folder_list ) )
{
    LIST *relation_mto1_folder_name_list(
        mto1_isa_related_folder_list );

    char *list_to_string(
        relation_mto1_folder_name_list(),
        ',' );

    ptr += sprintf( ptr,
        ",%s",
        list_to_string() );
}

if ( row_security_role_update_list_from )
{
    ptr += sprintf( ptr,
        "%s",
        row_security_role_update_list_from );
}

/* Usage */
LIST *query_primary_delimited_list(
    char *appaserver_table_name,
    LIST *primary_key_list,
    LIST *foreign_key_list,
    LIST *foreign_data_list );

/* Process */

/* Usage */
char *query_data_where(
    char *appaserver_table_name,
    LIST *where_attribute_name_list,
    LIST *where_attribute_data_list,
    LIST *folder_attribute_list );

/* Process */

/* Usage */
char *query_where_display(
    char *where_string,
    int max_length  );

/* Process */

/* Usage */
char *query_prompt_target_frame(
    char *datatype_name,
    pid_t process_id );

/* Process */

/* Usage */
unsigned long query_row_count(
    char *folder_name );

/* Process */
char *appaserver_table_name( folder_name );

char *query_row_count_system_string(
    char *appaserver_table_name() );

char *string_pipe_fetch(
    query_row_count_system_string() );

unsigned long strtoul(
    string_pipe_fetch(),
    NULL /* endptr */,
    10 /* base */);

/* Usage */
DICTIONARY *query_prompt_query_dictionary(
    const char *QUERY_RELATION_OPERATOR_PREFIX,
    const char *QUERY_EQUAL,
    DICTIONARY *prompt_dictionary );

/* Process */
DICTIONARY *query_dictionary = dictionary_small();

LIST *dictionary_key_list( prompt_dictionary );

for char *key in dictionary_key_list()
{
    char *dictionary_get(
        key,
        prompt_dictionary );

    dictionary_set(
        query_dictionary,
        key,
        dictionary_get() );

    char *query_relation_key(
        QUERY_RELATION_OPERATOR_PREFIX,
        key );

    dictionary_set(
        query_dictionary,
        strdup( query_relation_key() ),
        QUERY_EQUAL );
}
```

QUERY_FETCH

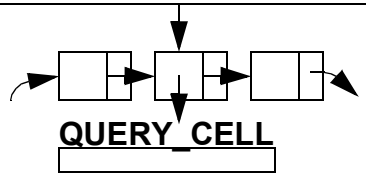
```
/* Usage */
QUERY_FETCH *query_fetch_new(
    char *folder_name,
    LIST *folder_attribute_append_isa_list,
    enum date_convert_format_enum
        destination_enum,
    LIST *query_select_name_list,
    char *query_system_string,
    boolean input_save_boolean );

/* Process */
QUERY_FETCH *query_fetch_calloc(
    void );

FILE *query_fetch_input_pipe(
    query_system_string );

for char input[] in
    string_input(
        query_fetch_input_pipe() )
{
    list_set(
        this->row_list,
        QUERY_ROW *query_row_parse(
            SQL_DELIMITER,
            folder_attribute_append_isa_list,
            destination_enum,
            query_select_name_list,
            input_save_boolean,
            input ) );
}

void pclose( query_fetch_input_pipe() );

/* Usage */
void query_fetch_html_display(
    char *query_fetch->folder_name,
    LIST *query_fetch->query_select_name_list,
    LIST *query_fetch->row_list );

/* Process */
char *query_fetch_display_system_string(
    folder_name,
    query_select_name_list );

FILE *appaserver_output_pipe(
    query_fetch_display_system_string() );

for QUERY_ROW *query_row in row_list
{
    fprintf(
        appaserver_output_pipe(),
        "%s\n",
        query_row->input );
}
pclose( appaserver_output_pipe() );

/* Usage */
char *query_fetch_display_system_string(
    char *folder_name,
    LIST *query_select_name_list );

/* Process */
char *string_initial_capital( folder_name );

char *list_to_string(
    query_select_name_list,
    ',' /* delimiter */ );

snprintf(
    system_string[],
    sizeof ( system_string ),
    "html_table.e \"%s\" \"%s\" '%c'",
    string_initial_capital() /* title */,
    list_to_string() /* heading_list */,
    SQL_DELIMITER );

/* Usage */
FILE *query_fetch_input_pipe(
    char *system_string );

/* Process */
FILE *appaserver_input_pipe( system_string );

/* Attributes */
char *folder_name,
LIST *query_select_name_list;
LIST *row_list;
```



QUERY_ROW (1)

```
/* Usage */
QUERY_ROW *query_row_parse(
    const char SQL_DELIMITER,
    LIST *folder_attribute_append_isa_list,
    enum date_convert_format_enum
        destination_enum,
    LIST *query_select_name_list(),
    boolean input_save_boolean,
    char *input /* stack memory */ );

/* Process */
LIST *cell_list = list_new();

int p = 0;

for char *attribute_name in query_select_name_list
{
    if ( !piece(
        char piece_buffer[],
        SQL_DELIMITER,
        input,
        p++ ) )
    {
        break;
    }

    list_set(
        cell_list,
        QUERY_CELL *query_cell_parse(
            folder_attribute_append_isa_list,
            destination_enum,
            attribute_name,
            strdup( piece_buffer )
                /* select_datum */ ) );
}

QUERY_ROW *query_row_new(
    input_save_boolean,
    input,
    cell_list );
```

```
/* Usage */
QUERY_ROW *query_row_new(
    boolean input_save_boolean,
    char *input /* stack memory or null */,
    LIST *cell_list );

/* Process */
QUERY_ROW *query_row_calloc(
    void );

if ( input_save_boolean ) strdup( input );

/* Attributes */
char *input;
LIST *cell_list;

/* Set externally */
boolean viewonly_boolean;
```



QUERY_ROW (2)

```
/* Usage */
LIST *query_row_delimited_list(
    const char ATTRIBUTE_MULTI_KEY_DELIMITER,
    LIST *query_fetch_row_list() );

/* Process */
LIST *delimited_list = list_new();

for QUERY_ROW *query_row in
    query_fetch_row_list
{
    list_set(
        delimited_list,
        char *query_cell_list_delimited_string(
            ATTRIBUTE_MULTI_KEY_DELIMITER,
            query_row->cell_list ) );
}

/* Usage */
void query_row_spreadsheet_output(
    FILE *output_file,
    QUERY_ROW *query_row );

/* Process */
char output_string[];
char *ptr = output_string;

for QUERY_CELL *query_cell in query_row->cell_list
{
    char *query_cell_quote_comma_string(
        query_cell,
        list_at_first( query_row->cell_list )
            /* first_cell_boolean */ );

    ptr +=
        sprintf(
            ptr,
            "%s",
            query_cell_quote_comma_string() );

    free( query_cell_quote_comma_string() );
}

void fprintf( output_file, "%s\n", output_string );

/* Usage */
void query_row_free(
    QUERY_ROW *query_row );

/* Process */
for QUERY_CELL *query_cell in query_row->cell_list
    void query_cell_free( query_cell );

/* Usage */
int query_row_list_days_range(
    char *date_attribute_name,
    LIST *row_list );

/* Process */
QUERY_ROW *first_row = list_first( row_list );

QUERY_ROW *last_row = list_last( row_list );

QUERY_CELL *first_cell =
    query_cell_seek(
        date_attribute_name,
        first_row->cell_list );

QUERY_CELL *last_cell =
    query_cell_seek(
        date_attribute_name,
        last_row->cell_list );

int date_days_between(
        first_cell->select_datum,
        last_cell->select_datum ) + 1;

/* Usage */
DICTIONARY *query_row_fetch_dictionary(
    LIST *folder_attribute_name_list,
    char *appaserver_table_name,
    char *query_cell_where_string() );

/* Process */

/* Usage */
QUERY_ROW *query_row_seek(
    LIST *row_list,
    char *date_string );

/* Process */
```

QUERY_CELL (1)

```
/* Usage */
QUERY_CELL *query_cell_parse(
    LIST *folder_attribute_list,
    enum date_convert_format_enum
        destination_enum,
    char *attribute_name,
    char *select_datum );

/* Process */
if ( list_length( folder_attribute_list ) )
{
    FOLDER_ATTRIBUTE *folder_attribute =
        folder_attribute_seek(
            (char *)0 /* folder_name */,
            attribute_name,
            folder_attribute_append_isa_list );

    if ( folder_attribute )
    {
        int primary_key_index =
            folder_attribute->
                primary_key_index;

        boolean attribute_is_date(
            folder_attribute->
                attribute->
                datatype_name );

        boolean attribute_is_date_time(
            folder_attribute->
                attribute->
                datatype_name );

        boolean attribute_is_current_date_time(
            folder_attribute->
                attribute->
                datatype_name );

        boolean attribute_is_number(
            folder_attribute->
                attribute->
                datatype_name );

        char *query_cell_display_datum(
            destination_enum,
            select_datum,
            attribute_is_date(),
            attribute_is_date_time(),
            attribute_is_current_date_time(),
            attribute_is_number() );
    }
}

QUERY_CELL *query_cell_new(
    attribute_name,
    select_datum,
    primary_key_index,
    attribute_is_date(),
    attribute_is_date_time(),
    attribute_is_current_date_time(),
    attribute_is_number(),
    query_cell_display_datum() );

/* Usage */
QUERY_CELL *query_cell_new(
    char *attribute_name,
    char *select_datum,
    int folder_attribute->primary_key_index,
    boolean attribute_is_date(),
    boolean attribute_is_date_time(),
    boolean attribute_is_current_date_time(),
    boolean attribute_is_number(),
    char *query_cell_display_datum() );

/* Process */
QUERY_CELL *query_cell_calloc(
    void );

/* Usage */
QUERY_CELL *query_cell_simple_new(
    char *attribute_name,
    char *select_datum );

/* Process */
QUERY_CELL *query_cell_new(
    attribute_name,
    select_datum,
    0 /* not primary_key_index */,
    0 /* not attribute_is_date */,
    0 /* not attribute_is_date_time */,
    0 /* not attribute_is_current_date_time */,
    0 /* not attribute_is_number */,
    (char *)0 /* query_cell_display_datum */ );

/* Usage */
char *query_cell_display_datum(
    enum date_convert_format_enum
        destination_enum,
    char *select_datum,
    boolean attribute_is_date(),
    boolean attribute_is_date_time(),
    boolean attribute_is_current_date_time(),
    boolean attribute_is_number() );

/* Process */
if (    attribute_is_date
||      attribute_is_date_time
||      attribute_is_current_date_time )
{
    DATE_CONVERT *date_convert =
        date_convert_new(
            date_convert_international
                /* source_enum */,
            destination_enum,
            select_datum
                /* source_date_string */ );

    char *display_datum =
        date_convert->
            return_date_string;
}
else
if ( attribute_is_number )
{
    char *display_datum =
        char *strdup(
            char *string_commas_number_string(
                select_datum ) );
}
else
{
    char *display_datum = select_datum;
}

/* Attributes */
char *attribute_name;
char *select_datum;
int primary_key_index;
boolean attribute_is_date;
boolean attribute_is_date_time;
boolean attribute_is_number;
char *display_datum;
```

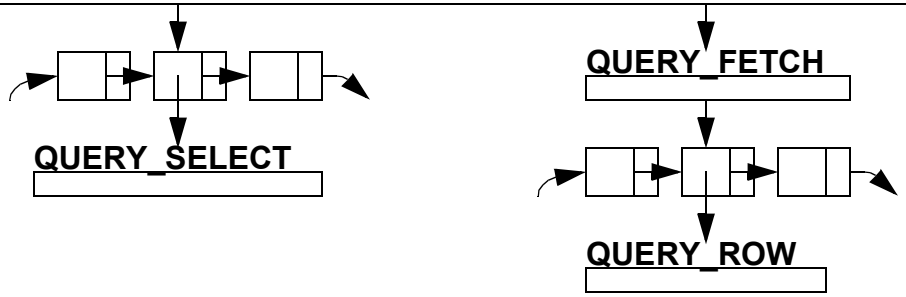
QUERY_CELL (2)

```
/* Usage */
LIST *query_cell_primary_data_list(
    LIST *query_row_cell_list );

/* Process */
LIST *primary_data_list = list_new();

for QUERY_CELL *query_cell in query_row_cell_list
{
    if ( query_cell->
            folder_attribute->
            primary_key_index )
    {
        list_set(
            primary_data_list,
            query_cell->select_datum );
    }
}

/* Usage */
char *query_cell_list_delimited_string(
    const char ATTRIBUTE_MULTI_KEY_DELIMITER,
    LIST *query_row_cell_list );

/* Process */
char delimited_string[];
char *ptr = delimited_string;

for QUERY_CELL *query_cell in query_row_cell_list
{
    if ( ptr != delimited_string )
            ptr +=
                sprintf(
                    ptr,
                    "%c",
                    ATTRIBUTE_MULTI_KEY_DELIMITER );

    ptr += sprintf(
            ptr,
            "%s",
            query_cell->select_datum );
}

/* Usage */
LIST *query_cell_attribute_data_list(
    LIST *attribute_name_list,
    LIST *query_row_cell_list );

/* Process */
LIST *data_list = list_new();

for char *attribute_name in attribute_name_list
{
    QUERY_CELL *query_cell =
        query_cell_seek(
            attribute_name,
            query_row_cell_list );

    list_set( data_list, query_cell->select_datum );
}

/* Usage */
LIST *query_cell_data_list(
    LIST *query_row_cell_list );

/* Process */
LIST *data_list = list_new();

for QUERY_CELL *query_cell in query_row_cell_list
{
    list_set( data_list, query_cell->select_datum );
}

/* Usage */
char *query_cell_quote_comma_string(
    QUERY_CELL *query_cell,
    boolean first_cell_boolean );

/* Process */
char quote_comma_string[];
char *ptr = quote_comma_string;

if ( !first_cell_boolean ) ptr += sprintf( ptr, "," );

if ( !query_cell->select_datum ) return strdup( "" );

if (    query_cell->folder_attribute
&&  query_cell->folder_attribute->attribute
&&  attribute_is_number(
            query_cell->
                folder_attribute->
                attribute->
                datatype_name ) )
{
    ptr += sprintf( ptr,
            "%s",
            query_cell->select_datum );
}
else
{
    ptr += sprintf( ptr,
            "\"%s\"",
             query_cell->display_datum );
}

/* Usage */
void query_cell_free(
    QUERY_CELL *query_cell );

/* Process */
if ( query_cell->display_datum )
    free( query_cell->display_datum );

if ( query_cell->select_datum
&& query_cell->select_datum !=
       query_cell->display_datum )
{
    free( query_cell->select_datum );
}

free( query_cell );

/* Usage */
char *query_cell_where_string(
    LIST *query_cell_list );

/* Process */
char where[];
char *ptr = where;

for QUERY_CELL *query_cell in query_cell_list
{
    if ( !query_cell->attribute_name ) return null;
    if ( !*query_cell->attribute_name ) return null;
    if ( !query_cell->select_datum ) return null;
    if ( !*query_cell->select_datum ) return null;

    if ( ptr != where) ptr += sprintf( ptr, " and " );

    ptr += sprintf( ptr,
            "%s = '%s'",
            query_cell->
                attribute_name,
            string_escape_quote(
                char destination[],
                query_cell->select_datum ) );
}
```

```
/* Usage */
void query_cell_command_line_replace(
    char *command_line,
    LIST *query_cell_list );

/* Process */
int string_instr( " ", command_line, 1 );

if ( string_instr() == -1 ) return;

command_line += string_instr();

for QUERY_CELL *query_cell in query_cell_list
{
    if ( !query_cell->attribute_name ) continue;

    string_with_space_search_replace(
        command_line
                /* source_destination */,
        query_cell->attribute_name
                /* search */,
        double_quotes_around(
            char destination[],
            query_cell->
                select_datum
                    /* replace */ ) );
}
/* Usage */
QUERY_CELL *query_cell_seek(
    char *attribute_name,
    LIST *cell_list );

/* Process */
for QUERY_CELL *query_cell in cell_list
{
    if ( string_strcmp(
        attribute_name,
        query_cell->attribute_name ) == 0 )
    {
        return query_cell;
    }
}
return NULL;

/* Usage */
char *query_cell_list_display(
    LIST *query_cell_list )

/* Process */
char display[];
char *ptr = display;

for QUERY_CELL *query_cell in query_cell_list
{
    if ( ptr != display ) ptr += sprintf( ptr, "," );

    ptr += sprintf( ptr,
        "%s=%s",
        query_cell->attribute_name,
        query_cell->select_datum );
}

/* Usage */
LIST *query_cell_attribute_build_list(
    LIST *key_list,
    LIST *data_list );

/* Process */

/* Usage */
/* Returns query_cell_list */
LIST *query_cell_attribute_list(
    LIST *attribute_name_list,
    LIST *query_row_cell_list );

/* Process */
LIST *query_cell_list = list_new();

for char *attribute_name in attribute_name_list
{
    QUERY_CELL *query_cell_seek(
        attribute_name,
        query_row_cell_list );

    if ( query_cell_seek() )
    {
        QUERY_CELL *query_cell_simple_new(
            query_cell_seek()->attribute_name,
            query_cell_seek()->select_datum );

        list_set( query_cell_list, query_cell_simple_new() );
    }
}

/* Usage */
boolean query_cell_list_set_attribute_name(
    LIST *attribute_name_list,
    LIST *query_cell_list /* in/out */ );

/* Process */
```

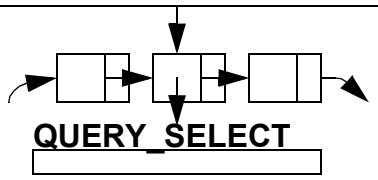
QUERY_CHOOSE_ISA

```
/* Usage */
QUERY_CHOOSE_ISA *query_choose_isa_new(
    char *application_name,
    char *login_name,
    char *one_isa_folder_name,
    LIST *folder_attribute_primary_list(),
    char *security_entity_where() );

/* Process */
QUERY_CHOOSE_ISA *query_choose_isa_calloc(
    void );

LIST *query_select_list(
    folder_attribute_primary_list );

char *query_select_list_string(
    query_select_list(),
    (LIST *)0 /* common_name_list */ );

LIST *query_select_name_list(
    query_select_list() );

char *query_order_string(
    query_select_name_list()
        /* folder_attribute_primary_key_list */,
    (LIST *)0 /* folder_attribute_name_list */,
    (DICTIONARY *)0 /* sort_dictionary */ );

char *query_system_string(
    application_name,
    query_select_list_string(),
    one_isa_folder_name /* from_string */,
    security_entity_where,
    (char *)0 /* group_by_string */,
    query_order_string(),
    0 /* max_rows */ );

enum date_convert_format_enum
    destination_enum =
        date_convert_login_name_enum(
            application_name,
            login_name );

QUERY_FETCH *query_fetch =
    query_fetch_new(
        one_isa_folder_name,
        folder_attribute_primary_list
            /* folder_attribute_append_isa_list */,
        destination_enum,
        query_select_name_list(),
        query_system_string(),
        0 /* not input_save_boolean */ );

LIST *query_row_delimited_list(
    SQL_DELIMITER,
    query_fetch->row_list );

/* Attributes */
LIST *query_select_list;
char *query_select_list_string;
LIST *query_select_name_list;
char *query_order_string;
char *query_system_string;
enum date_convert_format_enum destination_enum;
QUERY_FETCH *query_fetch;
LIST *query_row_delimited_list;
```



QUERY_SPREADSHEET

```
/* Usage */
QUERY_SPREADSHEET *query_spreadsheet_new(
    char *application_name,
    char *login_name,
    char *folder_name,
    char *security_entity_where,
    DICTIONARY *dictionary_separate->
        query_dictionary,
    LIST *dictionary_separate->no_display_name_list,
    LIST *role_attribute_exclude_lookup_name_list,
    LIST *folder_attribute_append_isa_list,
    LIST *folder_attribute_primary_key_list,
    LIST *relation_mto1_non_isa_list,
    LIST *relation_mto1_isa_list );

/* Process */
QUERY_SPREADSHEET *
    query_spreadsheet_calloc(
        void );

LIST *query_select_table_edit_list(
    application_name,
    folder_name,
    folder_attribute_append_isa_list,
    no_display_name_list,
    role_attribute_exclude_lookup_name_list,
    list_length( relation_mto1_isa_list ),
    (char *)0 /* attribute_not_null */ );

char *query_select_list_string(
    query_select_table_edit_list(),
    (LIST *)0 /* common_name_list */ );

LIST *query_select_name_list(
    query_select_table_edit_list() );

char *query_from_string(
    folder_name,
    relation_mto1_isa_list,
    (char *)0 /* row_security_role_update_list_from */ );

QUERY_TABLE_EDIT_WHERE *
    query_table_edit_where =
        query_table_edit_where_new(
            application_name,
            folder_name,
            relation_mto1_non_isa_list,
            relation_mto1_isa_list,
            folder_attribute_append_isa_list,
            security_entity_where,
            query_dictionary,
            (char *)0 /* attribute_not_null */,
            (char *)0 /* join_where */ );

char *query_order_string(
    folder_attribute_primary_key_list,
    (LIST *)0 /* folder_attribute_name_list */,
    (DICTIONARY *)0 /* sort_dictionary */ );

char *query_system_string(
    application_name,
    query_select_list_string(),
    query_from_string(),
    query_table_edit_where->string,
    (char *)0 /* group_by_string */,
    query_order_string(),
    QUERY_SPREADSHEET_MAX_ROWS );

enum date_convert_format_enum
    destination_enum =
        date_convert_login_name_enum(
            application_name,
            login_name );

/* Driver */
/* Returns row_count */
unsigned long query_spreadsheet_output(
    LIST *folder_attribute_append_isa_list,
    enum date_convert_format_enum
        query_spreadsheet->destination_enum,
    LIST *query_spreadsheet->query_select_name_list,
    char *query_spreadsheet->query_system_string,
    char *output_filename );

/* Process */
FILE *query_fetch_input_pipe(
    query_system_string );

FILE *query_spreadsheet_output_file(
    char *output_filename );

char *list_double_quote_comma_display(
    char destination[],
    query_select_name_list );

fprintf(  query_spreadsheet_output_file,
          "%s\n",
          list_double_quote_comma_display() );

for char input[] in string_input( query_fetch_input_pipe() )
{
    QUERY_ROW *query_row =
        query_row_parse(
            SQL_DELIMITER,
            folder_attribute_append_isa_list,
            destination_enum,
            query_select_name_list,
            0 /* not input_save_boolean */,
            input );

    void query_row_spreadsheet_output(
        query_spreadsheet_output_file(),
        query_row );

    void query_row_free( query_row );
}

void pclose( query_fetch_input_pipe() );
void fclose( query_spreadsheet_output_file() );

/* Attributes */
LIST *query_select_table_edit_list;
char *query_select_list_string;
LIST *query_select_name_list;
char *query_from_string;
QUERY_TABLE_EDIT_WHERE *
    query_table_edit_where;
char *query_order_string;
char *query_system_string;
enum date_convert_format_enum destination_enum;
```

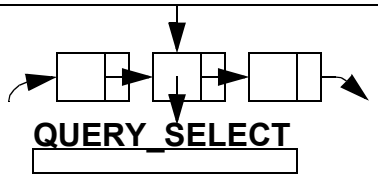


QUERY_SELECT

QUERY_TABLE_EDIT_WHERE

QUERY_TABLE_EDIT

```
/* Usage */
/* Note: frees each where. */
QUERY_TABLE_EDIT *query_table_edit_new(
    char *application_name,
    char *login_name,
    char *folder_name,
    char *security_entity_where,
    LIST *dictionary_separate->no_display_name_list,
    LIST *exclude_lookup_attribute_name_list,
    LIST *relation_mto1_non_isa_list,
    LIST *relation_mto1_isa_list,
    LIST *folder_attribute_append_isa_list,
    LIST *folder_attribute_primary_key_list,
    LIST *folder_attribute_name_list,
    DICTIONARY *query_dictionary,
    DICTIONARY *sort_dictionary,
    /* Null if not participating */
    ROW_SECURITY_ROLE_UPDATE_LIST *
        row_security_role_update_list,
    int TABLE_EDIT_QUERY_MAX_ROWS );

/* Process */
QUERY_TABLE_EDIT *query_table_edit_calloc(
    void );

LIST *query_select_table_edit_list(
    application_name,
    folder_name,
    folder_attribute_append_isa_list,
    no_display_name_list,
    exclude_lookup_attribute_name_list,
    list_length( relation_mto1_isa_list ),
    (row_security_role_update_list)
        ? row_security_role_update_list->
            attribute_not_null
        : (char *)0 );

char *query_select_list_string(
    query_select_table_edit_list(),
    (LIST *)0 /* common_name_list */ );

LIST *query_select_name_list(
    query_select_table_edit_list() );

char *query_from_string(
    folder_name,
    relation_mto1_isa_list,
    (row_security_role_update_list)
        ? row_security_role_update_list->from
        ? (char *)0 );

QUERY_TABLE_EDIT_WHERE *
    query_table_edit_where =
        query_table_edit_where_new(
            folder_name,
            relation_mto1_non_isa_list,
            relation_mto1_isa_list,
            folder_attribute_append_isa_list,
            security_entity_where,
            query_dictionary,
            (row_security_role_update_list)
                ? row_security_role_update_list->
                    attribute_not_null
                : (char *)0 );
            (row_security_role_update_list)
                ? row_security_role_update_list->
                    join_where
                : (char *)0 );

char *query_order_string(
    folder_attribute_primary_key_list,
    folder_attribute_name_list,
    sort_dictionary );

char *query_system_string(
    application_name,
    query_select_list_string(),
    query_from_string(),
    query_table_edit_where->string,
    (char *)0 /* group_by_string */,
    query_order_string(),
    table_edit_query_max_rows );

enum date_convert_format_enum
    destination_enum =
        date_convert_login_name_enum(
            application_name,
            login_name );

QUERY_FETCH *query_fetch =
    query_fetch_new(
        folder_name,
        folder_attribute_append_isa_list,
        destination_enum,
        query_select_name_list(),
        query_system_string(),
        0 /* not input_save_boolean */ );

/* Attributes */
LIST *query_select_table_edit_list;
char *query_select_list_string;
LIST *query_select_name_list;
char *query_from_string;
QUERY_TABLE_EDIT_WHERE *
    query_table_edit_where;
char *query_order_string;
char *query_system_string;
enum date_convert_enum destination_enum;
QUERY_FETCH *query_fetch;
```



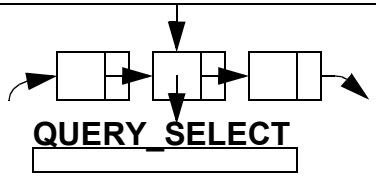
QUERY_TABLE_EDIT_WHERE

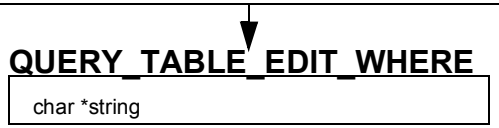
char *string

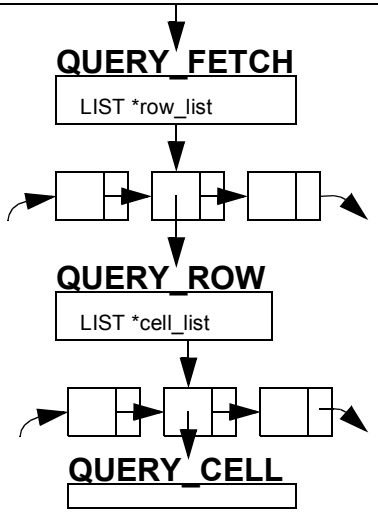
QUERY_FETCH

QUERY_PRIMARY_KEY

```
/* Usage */
/* Note: frees each where. */
QUERY_PRIMARY_KEY *query_primary_key_fetch(
    char *application_name,
    char *login_name,
    char *folder_name,
    /* Static memory or null */
    char *security_entity_where,
    LIST *relation_mto1_list,
    LIST *relation_mto1_isa_list,
    LIST *folder_attribute_append_isa_list,
    DICTIONARY *query_dictionary,
    char *additional_attribute_name );

/* Process */
QUERY_PRIMARY_KEY *query_primary_key_calloc(
    void );

int query_primary_key_isa_list_length(
    LIST *relation_mto1_isa_list );

LIST *query_select_primary_list(
    application_name,
    folder_name,
    folder_attribute_append_isa_list,
    additional_attribute_name,
    query_primary_key_isa_list_length()
        /* relation_mto1_isa_list_length */ );

char *query_select_list_string(
    query_select_primary_list(),
    (LIST *)0 /* common_name_list */ );

char *query_from_string(
    folder_name,
    relation_mto1_isa_list,
    (char *)0 /* row_security_role_update_list_from */ );

QUERY_TABLE_EDIT_WHERE *
    query_table_edit_where =
        query_table_edit_where_new(
            application_name,
            folder_name,
            relation_mto1_list,
            relation_mto1_isa_list,
            folder_attribute_append_isa_list,
            security_entity_where,
            query_dictionary,
            (char *)0 /* attribute_not_null */,
            (char *)0 /* join_where */ );

char *query_system_string(
    application_name,
    query_select_list_string(),
    query_from_string(),
    query_table_edit_where->string,
    (char *)0 /* group_by_string */,
    additional_attribute_name /* query_order_string */,
    0 /* max_rows */ );

LIST *query_select_name_list(
    query_select_primary_list() );

QUERY_FETCH *query_fetch =
    query_fetch_new(
        folder_name,
        folder_attribute_append_isa_list,
        date_convert_international
            /* destination_enum */,
        query_select_name_list(),
        query_system_string(),
        0 /* not input_save_boolean */ );

/* Attributes */
int isa_list_length;
LIST *query_select_primary_list;
char *query_select_list_string;
char *query_from_string;
QUERY_TABLE_EDIT_WHERE *
    query_table_edit_where;
char *query_system_string;
LIST *query_select_name_list;
QUERY_FETCH *query_fetch;
```

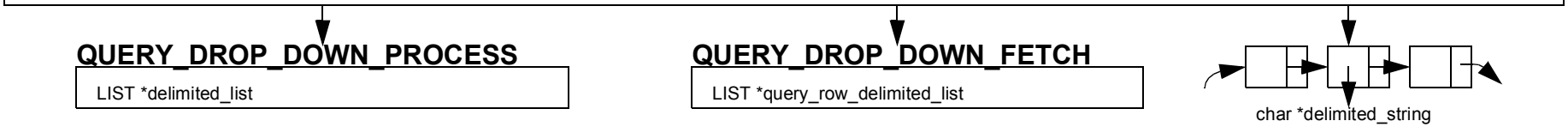






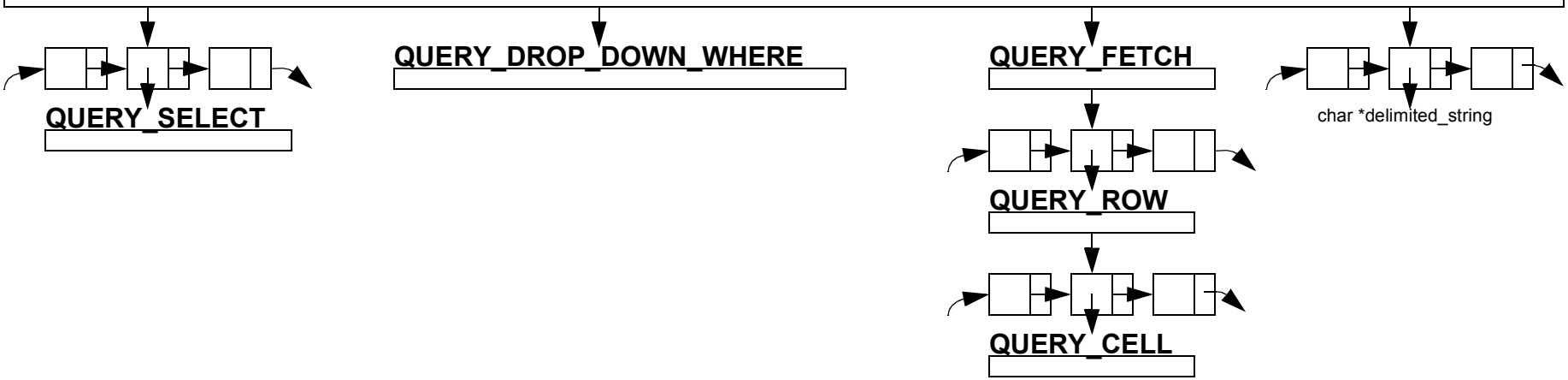
QUERY_DROP_DOWN

```
/* Usage */
QUERY_DROP_DOWN *query_drop_down_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *state,
    char *many_folder_name,
    char *one_folder_name,
    char *related_column,
    LIST *one_folder_attribute_list,
    char *populate_drop_down_process_name,
    LIST *relation_mto1_to_one_list,
    DICTIONARY *drop_down_dictionary,
    SECURITY_ENTITY *security_entity,
    LIST *common_name_list );

/* Process */
QUERY_DROP_DOWN *query_drop_down_calloc(
    void );

if ( populate_drop_down_process_name )
{
    QUERY_DROP_DOWN_PROCESS *
        query_drop_down_process =
            query_drop_down_process_new(
                application_name,
                session_key,
                login_name,
                role_name,
                state,
                many_folder_name,
                one_folder_name,
                related_column,
                one_folder_attribute_list,
                populate_drop_down_process_name,
                relation_mto1_to_one_list,
                drop_down_dictionary,
                security_entity );

    LIST *delimited_list =
        query_drop_down_process->
            delimited_list;
}
else
{
    QUERY_DROP_DOWN_FETCH *
        query_drop_down_fetch_new(
            application_name,
            login_name,
            one_folder_name,
            one_folder_attribute_list,
            relation_mto1_to_one_list,
            drop_down_dictionary,
            security_entity,
            common_name_list );

    LIST *delimited_list =
        query_drop_down_fetch->
            query_row_delimited_list;
}

/* Attributes */
QUERY_DROP_DOWN_PROCESS *
    query_drop_down_process;
QUERY_DROP_DOWN_FETCH *
    query_drop_down_fetch;
LIST *delimited_list;
```



QUERY_DROP_DOWN_FETCH

```
/* Usage */
QUERY_DROP_DOWN_FETCH *
   query_drop_down_fetch_new(
      char *application_name,
      char *login_name,
      char *one_folder_name,
      LIST *one_folder_attribute_list,
      LIST *relation_mto1_to_one_list,
      DICTIONARY *drop_down_dictionary,
      SECURITY_ENTITY *security_entity,
      LIST *common_name_list );

/* Process */
QUERY_DROP_DOWN_FETCH *
   query_drop_down_fetch_calloc(
      void );

LIST *folder_attribute_primary_list(
   char *one_folder_name,
   LIST *one_folder_attribute_list );

LIST *query_select_list(
   folder_attribute_primary_list() );

char *query_select_list_string(
   query_select_list(),
   common_name_list );

LIST *query_select_name_list(
   query_select_list() );

QUERY_DROP_DOWN_WHERE *
   query_drop_down_where =
      query_drop_down_where_new(
         application_name,
         one_folder_name,
         one_folder_attribute_list,
         relation_mto1_to_one_list,
         (security_entity)
            ? security_entity->where
            : (char *)0,
         drop_down_dictionary /* dictionary */ );

char *query_order_string(
   query_select_name_list(),
      /* folder_attribute_primary_key_list */,
   (LIST *)0 /* folder_attribute_name_list */,
   (DICTIONARY *)0 /* sort_dictionary */ );

char *query_system_string(
   application_name,
   query_select_list_string(),
   one_folder_name /* from_string */,
   query_drop_down_where->string,
   (char *)0 /* group_by_string */,
   query_order_string(),
   QUERY_DROP_DOWN_FETCH_MAX_ROWS );

enum date_convert_format_enum
   destination_enum =
      date_convert_login_name_enum(
         application_name,
         login_name );

QUERY_FETCH *query_fetch =
   query_fetch_new(
      one_folder_name,
      one_folder_attribute_list,
      destination_enum,
      query_select_name_list(),
      query_system_string(),
      0 /* not input_save_boolean */ );

LIST *query_row_delimited_list(
   ATTRIBUTE_MULTI_KEY_DELIMITER,
   query_fetch->row_list );

/* Attributes */
LIST *query_select_list;
char *query_select_list_string;
LIST *query_select_name_list;
QUERY_DROP_DOWN_WHERE *
   query_drop_down_where;
char *query_order_string;
char *query_system_string;
enum date_convert_enum destination_enum;
QUERY_FETCH *query_fetch;
LIST *query_row_delimited_list;
```

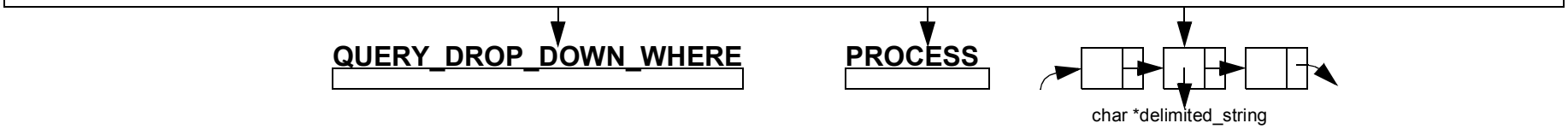


QUERY_DROP_DOWN_PROCESS (1)

```
/* Usage */
QUERY_DROP_DOWN_PROCESS *
   query_drop_down_process_new(
      char *application_name,
      char *session_key,
      char *login_name,
      char *role_name,
      char *state,
      char *many_folder_name,
      char *one_folder_name,
      char *related_column,
      LIST *one_folder_attribute_list,
      char *populate_drop_down_process_name,
      LIST *relation_mto1_to_one_list,
      DICTIONARY *dictionary,
      SECURITY_ENTITY *security_entity );

/* Process */
QUERY_DROP_DOWN_PROCESS *
   query_drop_down_process_calloc(
      void );

QUERY_DROP_DOWN_WHERE *
   query_drop_down_where =
      query_drop_down_where_new(
         application_name,
         one_folder_name,
         one_folder_attribute_list,
         relation_mto1_to_one_list,
         (security_entity)
            ? security_entity->where
            : (char *)0,
         dictionary );

PROCESS *process =
   process_fetch(
      populate_drop_down_process_name,
      (char *)0 /* document_root */,
      (char *)0 /* relative_source_directory */,
      1 /* check_executable_inside_filesystem */,
      appaserver_parameter_mount_point() );

char *appaserver_error_filename(
      application_name );

char *query_drop_down_process_command_line(
   application_name,
   session_key,
   login_name,
   role_name,
   state,
   many_folder_name,
   one_folder_name,
   related_column,
   populate_drop_down_process_name,
   dictionary,
   query_drop_down_where->string,
   process->command_line,
   appaserver_error_filename() );

LIST *delimited_list =
   list_pipe_fetch(
      query_drop_down_process_command_line() );
```

QUERY_DROP_DOWN_PROCESS (2)

```
/* Usage */
char *query_drop_down_process_command_line(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *state,
    char *many_folder_name,
    char *one_folder_name,
    char *related_column,
    char *populate_drop_down_process_name,
    DICTIONARY *dictionary,
    char *where_string,
    char *process_command_line,
    char *appaserver_error_filespecification );

/* Process */
char *process_replace_command_line(
    UPDATE_PREUPDATE_PREFIX,
    application_name,
    session_key,
    login_name,
    role_name,
    (char *)0 /* folder_name */,
    (char *)0 /* target_frame */,
    state,
    populate_drop_down_process_name,
    many_folder_name,
    one_folder_name,
    related_column,
    (char *)0 /* update_results_string */,
    (char *)0 /* update_error_string */,
    dictionary /* operation_row_list_dictionary */,
    dictionary /* dictionary_single_row */,
    where_string,
    0 /* row_number */,
    0 /* row_count */,
    0 /* parent_process_id */,
    (LIST *)0 /* primary_key_data_list */,
    (char *)0 /* execute_yn */,
    (LIST *)0 /* insert_datum_list */,
    (LIST *)0 /* update_attribute_list */,
    (LIST *)0 /* query_cell_list */,
    appaserver_error_filespecification,
    process_command_line /* input_command_line */ );

/* Attributes */
QUERY_DROP_DOWN_WHERE *
    query_drop_down_where;
PROCESS *process;
char *appaserver_error_filename;
char *command_line;
LIST *delimited_list;
```



QUERY_TABLE_EDIT_WHERE

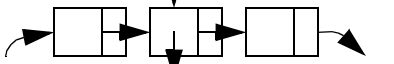
```
/* Usage */
/* Note: frees each where */
QUERY_TABLE_EDIT_WHERE *
   query_table_edit_where_new(
      char *folder_name,
      LIST *relation_mto1_list(),
      LIST *relation_mto1_isa_list(),
      LIST *folder_attribute_append_isa_list(),
      /* Static memory or null */
      char *security_entity_where(),
      DICTIONARY *query_dictionary,
      char *attribute_not_null,
      char *row_security_role_update_list_join_where );

/* Process */
QUERY_TABLE_EDIT_WHERE *
   query_table_edit_where_calloc(
      void );

int relation_mto1_isa_list_length =
   list_length( relation_mto1_isa_list );

char *query_table_name(
   folder_name,
   relation_mto1_isa_list_length,
   attribute_not_null );

QUERY_DROP_DOWN_LIST *
   query_drop_down_list =
      query_drop_down_list_new(
         relation_mto1_list
            /* relation_mto1_to_one_list */,
         query_dictionary /* dictionary */,
         query_table_name() );

QUERY_ISA_DROP_DOWN_LIST *
   query_isq_drop_down_list =
      query_isa_drop_down_list_new(
         relation_mto1_isa_list,
         query_dictionary /* dictionary */ );

char *query_table_edit_where_drop_down_where(
   char *query_drop_down_list->
      query_drop_down_relation_where,
   char *query_isa_drop_down_list->where );

QUERY_ATTRIBUTE_LIST *query_attribute_list =
   query_attribute_list_new(
      folder_attribute_append_isa_list,
      query_dictionary /* dictionary */,
      list_length(
         relation_mto1_isa_list )
         /* relation_mto1_isa_list_length */,
      attribute_not_null );

QUERY_ISA *query_isa =
   query_isa_new(
      application_name,
      relation_mto1_isa_list );

/* Note: frees each non-static where */
char *query_table_edit_where_string(
   /* Static memory or null */
   char *security_entity_where,
   char *query_table_edit_where_drop_down_where()
      /* query_drop_down_relation_where */,
   char *query_attribute_list->where,
   char *query_isa->join_where,
   char *row_security_role_update_list_join_where );

/* Usage */
void query_table_edit_where_free(
   QUERY_TABLE_EDIT_WHERE *
      query_table_edit_where );

/* Process */
string_free(
   query_table_edit_where->
      relation_mto1_query_drop_down_list->
      query_drop_down_relation_where );

string_free(
   query_table_edit_where->
      relation_mto1_isa_query_drop_down_list->
      query_drop_down_relation_where );

string_free(
   query_table_edit_where->
      drop_down_where );

string_free( query_table_edit_where->string );
free( query_table_edit_where );

/* Attributes */
int relation_mto1_isa_list_length;
char *query_table_name;
QUERY_DROP_DOWN_LIST *
   query_drop_down_list;
QUERY_ISA_DROP_DOWN_LIST *
   query_isa_drop_down_list;
char *drop_down_where;
QUERY_ATTRIBUTE_LIST *query_attribute_list;
QUERY_ISA *query_isa;
char *string;
```

QUERY_DROP_DOWN_LIST

```
char *query_drop_down_relation_where
LIST *query_drop_down_relation_attribute_name_list
```

QUERY_ISA_DROP_DOWN_LIST

```
char *where
```



QUERY_DROP_DOWN_LIST

QUERY_ISA

```
char *join_where
```

QUERY_ATTRIBUTE_LIST

```
char *where
```

QUERY_DROP_DOWN_WHERE

```
/* Usage */
QUERY_DROP_DOWN_WHERE *
   query_drop_down_where_new(
      char *one_folder_name,
      LIST *one_folder_attribute_list,
      LIST *relation_mto1_to_one_list,
      char *security_entity_where,
      DICTIONARY *dictionary );

/* Process */
QUERY_DROP_DOWN_WHERE *
   query_drop_down_where_calloc(
      void );

char *query_table_name(
   one_folder_name,
   0 /* relation_mto1_isa_list_length */,
   (char *)0 /* attribute_not_null */ );

QUERY_DROP_DOWN_LIST *
   query_drop_down_list =
      query_drop_down_list_new(
         relation_mto1_to_one_list,
         dictionary,
         query_table_name()
            /* many_table_name */ );

QUERY_ATTRIBUTE_LIST *query_attribute_list =
   query_attribute_list_new(
      one_folder_attribute_list
         /* folder_attribute_append_isa_list */,
      dictionary,
      0 /* relation_mto1_isa_list_length */,
      (char *)0 /* attribute_not_null */ );

char *query_drop_down_where_string(
   char *security_entity_where,
   char *query_drop_down_list->
      query_drop_down_relation_where,
   char *query_attribute_list->where );

/* Attributes */
char *query_table_name;
QUERY_DROP_DOWN_LIST *query_drop_down_list;
QUERY_ATTRIBUTE_LIST *query_attribute_list;
char *string;
```

QUERY_JOIN_ONE_TO_MANY_WHERE

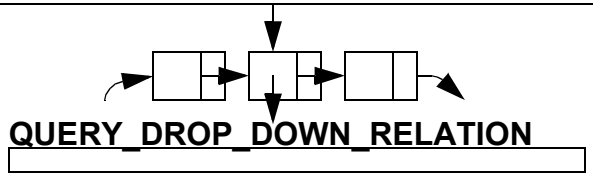
```
/* Usage */
QUERY_JOIN_ONE_TO_MANY_WHERE *
    query_join_one_to_many_where_new(
        char SQL_DELIMITER,
        char *primary_delimited_string,
        LIST *relation_join_one2m->foreign_key_list );

/* Process */
QUERY_JOIN_ONE_TO_MANY_WHERE *
    query_join_one_to_many_where_calloc(
        void );

char *query_join_one_to_many_where_string(
    SQL_DELIMITER,
    primary_delimited_string,
    foreign_key_list );
```

```
/* Usage */
char *query_join_one_to_many_where_string(
    char SQL_DELIMITER,
    char *primary_delimited_string,
    LIST *foreign_key_list );

/* Process */
char string[];
char *ptr = string;

for char *foreign_key in foreign_key_list
{
    piece(
        buffer[],
        SQL_DELIMITER,
        primary_delimited_string,
        p++ );
```

```
    if ( ptr != string ) ptr += sprintf( ptr, " and " );

    ptr +=
        sprintf(
            ptr,
            "%s = '%s'",
            foreign_key,
            buffer );
}

char *strdup( string );

/* Attributes */
char *string;
```

QUERY_DROP_DOWN_LIST

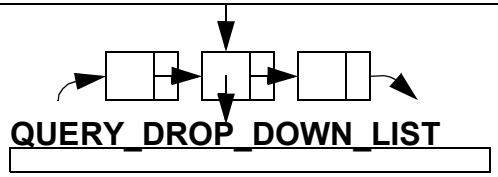
```
/* Usage */
QUERY_DROP_DOWN_LIST *
    query_drop_down_list_new(
        LIST *relation_mto1_to_one_list,
        DICTIONARY *dictionary,
        char *many_table_name
            /* optional */ );

/* Process */
QUERY_DROP_DOWN_LIST *
    query_drop_down_list_calloc(
        void );

int dictionary_highest_index(
    dictionary );

LIST *query_drop_down_relation_list = list_new();

for RELATION_MTO1 *relation_mto1 in
    relation_mto1_to_one_list
{
    if ( relation_mto1->relation->relation_type_isa )
            continue;

    QUERY_DROP_DOWN_RELATION *
        query_drop_down_relation =
            query_drop_down_relation_new(
                dictionary,
                many_table_name,
                dictionary_highest_index(),
                relation_mto1 );

    if ( query_drop_down_relation )
    {
        list_set(
            query_drop_down_relation_list,
            query_drop_down_relation );
    }
}

char *query_drop_down_relation_where(
    query_drop_down_relation_list );

LIST *query_drop_down_relation_attribute_name_list(
    query_drop_down_relation_list );

/* Attributes */
int dictionary_highest_index;
LIST *query_drop_down_relation_list;
char *query_drop_down_relation_where;
LIST *query_drop_down_relation_attribute_name_list;
```



QUERY_DROP_DOWN_RELATION

QUERY_ISA_DROP_DOWN_LIST

```
/* Usage */
QUERY_ISA_DROP_DOWN_LIST *
   query_isa_drop_down_list_new(
      LIST *relation_mto1_isa_list,
      DICTIONARY *dictionary );

/* Process */
QUERY_ISA_DROP_DOWN_LIST *
   query_isa_drop_down_list_calloc(
      void );

LIST *list = list_new();

for RELATION_MTO1 *relation_mto1 in
   relation_mto1_isa_list
{
   char *appaserver_table_name(
      relation_mto1->one_folder_name );

   QUERY_DROP_DOWN_LIST *
      query_drop_down_list_new(
         relation_mto1->relation_mto1_list
            /* relation_mto1_to_one_list */,
         dictionary,
         appaserver_table_name()
            /* many_table_name */ );

   if ( query_drop_down_list_new() )
   {
      list_set(
         list,
         query_drop_down_list_new() );
   }
}

if ( list_length( list ) )
{
   char *query_isa_drop_down_list_where(
      list /* query_drop_down_list_list */ );
}

/* Usage */
char *query_isa_drop_down_list_where(
   LIST *query_drop_down_list_list );

/* Process */
char where[];
char *ptr = where;

for QUERY_DROP_DOWN_LIST *query_drop_down_list in
   query_drop_down_list_list
{
   if ( !list_at_first( query_drop_down_list_list ) )
         ptr += sprintf( ptr, " and " );

   ptr += sprintf( ptr,
      "%s",
      query_drop_down_list->
         query_drop_down_relation_where );
}

/* Attributes */
LIST *list;
char *where;
```

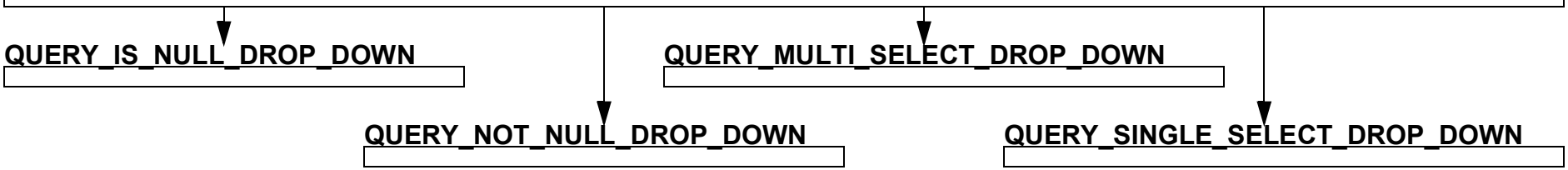


QUERY_DROP_DOWN_RELATION (1)

```
/* Usage */
QUERY_DROP_DOWN_RELATION *
    query_drop_down_relation_new(
        DICTIONARY *dictionary,
        char *many_table_name
            /* optional */,
        int dictionary_highest_index(),
        RELATION_MTO1 *relation_mto1 );

/* Process */
QUERY_DROP_DOWN_RELATION *
    query_drop_down_relation_calloc(
        void );

int dictionary_key_list_highest_index(
    ATTRIBUTE_MULTI_KEY_DELIMITER,
    relation_mto1->relation_foreign_key_list,
    dictionary,
    dictionary_highest_index );

if ( dictionary_key_list_highest_index() < 0 ) return NULL;

if ( query_is_null_drop_down_boolean(
        dictionary,
        relation_mto1->relation_foreign_key_list,
        dictionary_key_list_highest_index() ) )
{
    QUERY_IS_NULL_DROP_DOWN *
        query_is_null_drop_down =
            query_is_null_drop_down_new(
                many_table_name,
                relation_mto1->relation_foreign_key_list ) );
}
else
if ( query_not_null_drop_down_boolean(
        dictionary,
        relation_mto1->relation_foreign_key_list,
        dictionary_key_list_highest_index() ) )
{
    QUERY_NOT_NULL_DROP_DOWN *
        query_not_null_drop_down =
            query_not_null_drop_down_new(
                many_table_name,
                relation_mto1->relation_foreign_key_list ) );
}
else
if ( dictionary_key_list_highest_index() >= 1 )
{
    QUERY_MULTI_SELECT_DROP_DOWN *
        query_multi_select_drop_down =
            query_multi_select_drop_down_new(
                relation_mto1->many_folder_name,
                relation_mto1->one_folder_name,
                many_table_name,
                dictionary,
                relation_mto1->relation_foreign_key_list,
                dictionary_key_list_highest_index() );
}
else
{
    QUERY_SINGLE_SELECT_DROP_DOWN *
        query_single_select_drop_down =
            query_single_select_drop_down_new(
                many_table_name,
                dictionary,
                relation_mto1->relation_foreign_key_list );

    if ( !query_single_select_drop_down )
    {
        query_drop_down_relation = NULL;
    }
}

/* Usage */
/* Note: it frees char *where */
char *query_drop_down_relation_where(
    LIST *query_drop_down_relation_list );

/* Process */
```

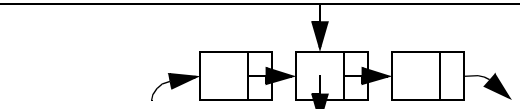
QUERY_DROP_DOWN_RELATION (2)

```
/* Usage */
LIST *query_drop_down_relation_attribute_name_list(
    LIST *query_drop_down_relation_list );

/* Process */
LIST *attribute_name_list = list_new();

for QUERY_DROP_DOWN_RELATION *
    query_drop_down_relation
        in query_drop_down_relation_list
{
    if ( query_drop_down_relation->
            query_is_null_drop_down )
    {
        list_set_list(
            attribute_name_list,
            query_drop_down_relation->
                query_is_null_drop_down->
                foreign_key_list );
    }
    else
    if ( query_drop_down_relation->
            query_not_null_drop_down )
    {
        list_set_list(
            attribute_name_list,
            query_drop_down_relation->
                query_not_null_drop_down->
                foreign_key_list );
    }
    else
    if ( query_drop_down_relation->
            query_multi_select_drop_down )
    {
        list_set_list(
            attribute_name_list,
            query_drop_down_relation->
                query_multi_select_drop_down->
                foreign_key_list );
    }
    else
    if ( query_drop_down_relation->
            query_single_select_drop_down )
    {
        list_set_list(
            attribute_name_list,
            query_drop_down_relation->
                query_single_select_drop_down->
                foreign_key_list );
    }
}

/* Attributes */
char *many_folder_name;
char *one_folder_name;
int dictionary_key_list_highest_index;
QUERY_IS_NULL_DROP_DOWN *
    query_is_null_drop_down;
QUERY_NOT_NULL_DROP_DOWN*
    query_not_null_drop_down;
QUERY_MULTI_SELECT_DROP_DOWN *
    query_multi_select_drop_down;
QUERY_SINGLE_SELECT_DROP_DOWN *
    query_single_select_drop_down;
```



QUERY_MULTI_SELECT_DROP_DOWN

```
/* Usage */
QUERY_MULTI_SELECT_DROP_DOWN *
    query_multi_select_drop_down_new(
        char *many_folder_name,
        char *one_folder_name,
        char *many_table_name /* optional */,
        DICTIONARY *dictionary,
        LIST *foreign_key_list,
        int highest_index );

/* Process */
if ( highest_index < 1 ) exit( 1 );

QUERY_MULTI_SELECT_DROP_DOWN *
    query_multi_select_drop_down_calloc(
        void );

LIST *query_multi_drop_down_column_list = list_new();

for index = 1 to highest_index
{
    for char *foreign_key in foreign_key_list
    {
        char *query_multi_select_drop_down_key(
            foreign_key,
            index );

        if ( ( dictionary_get(
                query_multi_select_drop_down_key(),
                dictionary ) ) )
        {
            QUERY_MULTI_DROP_DOWN_COLUMN *
                /* Returns null if exists in list already */
                query_multi_drop_down_column_new(
                    many_folder_name,
                    one_folder_name,
                    many_table_name,
                    query_multi_drop_down_column_list,
                    foreign_key,
                    dictionary_get()
                        /* datum */ );

            if ( query_multi_drop_down_column_new() )
            {
                list_set(
                    query_multi_drop_down_column_list,
                    query_multi_drop_down_column_new() );
            }
        }
    }
}

if ( !list_length(query_multi_drop_down_column_list ) )
    exit( 1 );

char *query_multi_drop_down_column_where(
    query_multi_drop_down_column_list );

/* Usage */
char *query_multi_select_drop_down_key(
    char *foreign_key,
    int index );

/* Process */
sprintf(
    static key[],
    "%s_%d",
    foreign_key,
    index );

/* Attributes */
char *many_folder_name;
char *one_folder_name;
LIST *foreign_key_list;
LIST *query_multi_drop_down_column_list;
char *query_multi_drop_down_column_where;
```



QUERY_MULTI_DROP_DOWN_COLUMN

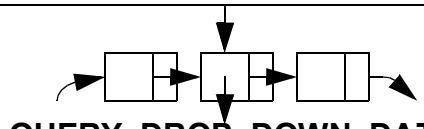
QUERY_SINGLE_SELECT_DROP_DOWN

```
/* Usage */
QUERY_SINGLE_SELECT_DROP_DOWN *
   query_single_select_drop_down_new(
      char *many_table_name,
      DICTIONARY *dictionary,
      LIST *foreign_key_list );

/* Process */
boolean query_single_select_drop_down_boolean(
   dictionary,
   foreign_key_list );

if ( !query_single_select_drop_down_boolean() ) return null;

QUERY_SINGLE_SELECT_DROP_DOWN *
   query_single_select_drop_down_calloc(
      void );

char *query_drop_down_key(
   foreign_key_list,
   0 /* index */,
   ATTRIBUTE_MULTI_KEY_DELIMITER );

char *data_list_string =
   dictionary_get(
      query_drop_down_key(),
      dictionary ) );

if ( data_list_string && *data_list_string )
{
   QUERY_DROP_DOWN_ROW *
      query_drop_down_row =
         query_drop_down_row_new(
            ATTRIBUTE_MULTI_KEY_DELIMITER,
            many_table_name,
            foreign_key_list,
            data_list_string ) );
}
else
{
   return null;
}

char *query_single_select_drop_down_where(
   this );

/* Usage */
char *query_single_select_drop_down_where(
   QUERY_SINGLE_SELECT_DROP_DOWN *
      query_single_select_drop_down );

/* Process */
return
   query_single_select_drop_down->
      query_drop_down_row->
      query_drop_down_datum_list_where;

/* Usage */
boolean query_single_select_drop_down_boolean(
   DICTIONARY *dictionary,
   LIST *foreign_key_list );

/* Process */
char *list_delimited_string(
   foreign_key_list,
   ATTRIBUTE_MULTI_KEY_DELIMITER );

char *query_relation_key(
   QUERY_RELATION_OPERATOR_PREFIX,
   list_delimited_string() /* attribute_name */ );

free( list_delimited_string() );

char *dictionary_get(
   query_relation_key(),
   dictionary );

if ( dictionary_get() )
   drop_down_boolean = 0;
else
   drop_down_boolean = 1;

/* Attributes */
LIST *foreign_key_list;
char *data_list_string;
QUERY_DROP_DOWN_ROW *query_drop_down_row;
char *where;
```

QUERY_DROP_DOWN_ROW

QUERY_DICTIONARY

```
/* Usage */
QUERY_DICTIONARY *query_dictionary_fetch(
    char *application_name,
    DICTIONARY *dictionary,
    char *folder_name,
    LIST *select_attribute_name_list,
    LIST *where_attribute_name_list );

/* Process */
QUERY_DICTIONARY *query_dictionary_calloc(
            void );

char *query_dictionary_select_string(
    LIST *select_attribute_name_list );

LIST *query_drop_down_datum_list = list_new();

for char *attribute_name in
    where_attribute_name_list
{
    char *dictionary_get(
        attribute_name,
        dictionary );

    if ( !dictionary_get() ) exit( 1 );

    QUERY_DROP_DOWN_DATUM *
        query_drop_down_datum =
            query_drop_down_datum_new(
                (char *)0 /* many_table_name */,
                attribute_name /* foreign_key */,
            dictionary_get() /* datum */ );

    list_set(
        query_drop_down_datum_list,
        query_drop_down_datum );
}

char *query_drop_down_datum_list_where(
    query_drop_down_datum_list,
        "and" /* conjunction */ );

char *appaserver_table_name(
    folder_name );

char *query_dictionary_system_string(
    char *query_dictionary_select_string(),
    char *appaserver_table_name(),
    char *query_drop_down_datum_list_where() );

LIST *delimited_list =
    LIST *list_pipe_fetch(
        query_dictionary_system_string() );

/* Attributes */
char *select_string;
LIST *query_drop_down_datum_list;
char *query_drop_down_datum_list_where;
char *system_string;
LIST *delimited_list;
```

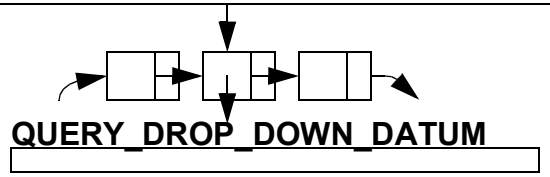


QUERY_DROP_DOWN_DATUM

```
char *where
```

QUERY_MULTI_DROP_DOWN_COLUMN (1)

```
/* Usage */
QUERY_MULTI_DROP_DOWN_COLUMN *
   /* Returns null if exists in list already */
   query_multi_drop_down_column_new(
      char *many_folder_name,
      char *one_folder_name,
      char *many_table_name,
      LIST *query_multi_drop_down_column_list,
      char *foreign_key,
      char *datum );

/* Process */
QUERY_MULTI_DROP_DOWN_COLUMN *
   query_multi_drop_down_column_seek(
      one_folder_name,
      foreign_key,
      query_multi_drop_down_column_list );

if ( query_multi_drop_down_column_seek() )
{
   list_set(
      query_drop_down_datum_list,
      query_drop_down_datum_new(
         many_table_name,
         foreign_key,
         datum ) );
   return NULL;
}

QUERY_MULTI_DROP_DOWN_COLUMN *
   query_multi_drop_down_column_calloc(
         void );

LIST *query_drop_down_datum_list = list_new();

list_set(
   query_drop_down_datum_list,
   query_drop_down_datum_new(
      many_table_name,
      foreign_key,
      datum ) );

/* Usage */
QUERY_MULTI_DROP_DOWN_COLUMN *
   query_multi_drop_down_column_seek(
      char *one_folder_name,
      char *foreign_key,
      LIST *query_multi_drop_down_column_list );

/* Process */
for QUERY_MULTI_DROP_DOWN_COLUMN *
   query_multi_drop_down_column in
      query_multi_drop_down_column_list
{
   if ( strcmp(
         query_multi_drop_down_column->
            one_folder_name,
         one_folder_name ) != 0 )
   {
      continue;
   }

   if ( strcmp(
         query_multi_drop_down_column->
            foreign_key,
         foreign_key ) != 0 )
   {
      continue;
   }
   return query_multi_drop_down_column;
}
return NULL;

/* Attributes */
char *many_folder_name;
char *one_folder_name;
char *foreign_key;
LIST *query_drop_down_datum_list;
```

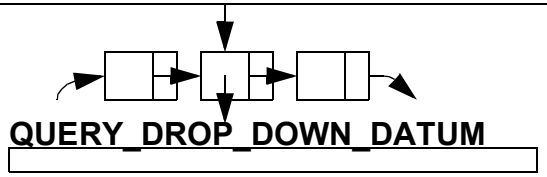


QUERY_MULTI_DROP_DOWN_COLUMN (2)

```
/* Usage */
char *query_multi_drop_down_column_where(
    LIST *query_multi_drop_down_column_list );

/* Process */
char where[ STRING_WHERE_BUFFER ];
char *ptr = where;
boolean first_time = 1;

ptr += sprintf( ptr, "(" );

for QUERY_MULTI_DROP_DOWN_COLUMN *
    query_multi_drop_down_column in
        query_multi_drop_down_column_list
{
    if ( !first_time )
        ptr += sprintf( ptr, " and " );
    else
        first_time = 0;

    char *query_drop_down_datum_list_where(
        query_multi_drop_down_column->
            query_drop_down_datum_list,
            "or" /* conjunction */ );

    ptr +=
        sprintf(
            ptr,
            "%s",
            query_drop_down_datum_list_where() );

    free( query_drop_down_datum_list_where() );
}

ptr += sprintf( ptr, ")" );
```

QUERY_DROP_DOWN_ROW

```
/* Usage */
QUERY_DROP_DOWN_ROW *
   query_drop_down_row_new(
      const char MULTI_ATTRIBUTE_KEY_DELIMITER;
      char *many_table_name,
      LIST *foreign_key_list,
      char *data_list_string );

/* Process */
QUERY_DROP_DOWN_ROW *
   query_drop_down_row_calloc(
         void );

int p = 0;

LIST *query_drop_down_datum_list = list_new();

for char *foreign_key in foreign_key_list()
{
   piece(
      char datum[],
      MULTI_ATTRIBUTE_KEY_DELIMITER,
      data_list_string,
      p++ );

   list_set(
      query_drop_down_datum_list,
      query_drop_down_datum_new(
            many_table_name,
            foreign_key,
            datum ) );
}

char *query_drop_down_datum_list_where(
   query_drop_down_datum_list,
   "and" /* conjunction */ );

/* Attributes */
char *data_list_string;
LIST *query_drop_down_datum_list;
char *query_drop_down_datum_list_where;
```



QUERY_DROP_DOWN_DATUM

```
/* Usage */
QUERY_DROP_DOWN_DATUM *
    query_drop_down_data_new(
        char *many_table_name,
        char *foreign_key,
        char *datum );

/* Process */
QUERY_DROP_DOWN_DATUM *
    query_drop_down_datum_calloc(
        void );

char *query_drop_down_datum_where(
    many_table_name,
    foreign_key,
    datum );

/* Usage */
char *query_drop_down_datum_where(
    char *many_table_name,
    char *foreign_key,
    char *datum );

/* Process */
sprintf(
        char where[],
        "%s = '%s'",
        char *attribute_full_attribute_name(
            many_table_name,
            foreign_key ),
        datum );

/* Usage */
char *query_drop_down_datum_list_where(
    LIST *query_drop_down_datum_list,
    char *conjunction );

/* Process */
char where[];
char *ptr = where;
boolean first_time = 1;

ptr += sprintf( ptr, "(" );

for QUERY_DROP_DOWN_DATUM *
    query_drop_down_datum in
        query_drop_down_datum_list
{
    if ( first_time )
    {
        first_time = 0;
    }
    else
    {
        ptr +=
            sprintf(
                ptr,
                " %s ",
                conjunction );
    }

    ptr +=
        sprintf(
            ptr,
            "%s",
            query_drop_down_datum->where );

    free( query_drop_down_datum->where );
}

ptr += sprintf( ptr, ")" );

/* Attribute */
char *where;
```

QUERY_IS_NULL_DROP_DOWN

```
/* Usage */
QUERY_IS_NULL_DROP_DOWN *
    query_is_null_drop_down_new(
        char *many_table_name,
        LIST *foreign_key_list );

/* Process */
QUERY_IS_NULL_DROP_DOWN *
    query_is_null_drop_down_calloc(
        void );

char *query_is_null_drop_down_where(
    many_table_name,
    foreign_key_list );

/* Usage */
char *query_is_null_drop_down_where(
    char *many_table_name,
    LIST *foreign_key_list );

/* Process */
char where[];
char *ptr = where;

for char *foreign_key in foreign_key_list
{
    if ( ptr != where ) ptr += sprintf( ptr, " and " );

    ptr +=
        sprintf(
            ptr,
            "%s is null",
            attribute_full_attribute_name(
                many_table_name,
                foreign_key ) );
}

char *strdup( where );

/* Usage */
boolean query_is_null_drop_down_boolean(
    DICTIONARY *dictionary,
    LIST *foreign_key_list,
    int highest_index );

/* Process */
for (   int index = 0;
        index <= highest_index;
        index++ )
{
    char *query_drop_down_key(
        foreign_key_list,
        index,
        MULTI_ATTRIBUTE_KEY_DELIMITER );

    char *dictionary_get(
        query_drop_down_key(),
        dictionary );

    if ( string_strcmp(
        dictionary_get(),
        QUERY_IS_NULL ) == 0 )
    {
        return 1;
    }
}
return 0;

/* Attribute */
LIST *foreign_key_list;
char *where;
```

QUERY_NOT_NULL_DROP_DOWN

```
/* Usage */
QUERY_NOT_NULL_DROP_DOWN *
    query_not_null_drop_down_new(
        char *many_table_name,
        LIST *foreign_key_list );

/* Process */
QUERY_NOT_NULL_DROP_DOWN *
    query_not_null_drop_down_calloc(
        void );

char *query_not_null_drop_down_where(
    many_table_name,
    foreign_key_list );

/* Usage */
char *query_not_null_drop_down_where(
    char *many_table_name,
    LIST *foreign_key_list );

/* Process */
char where[];
char *ptr = where;

for char *foreign_key in foreign_key_list
{
    if ( ptr != where ) ptr += sprintf( ptr, " and " );

    ptr +=
        sprintf(
            ptr,
            "%s is not null",
            attribute_full_attribute_name(
                many_table_name,
                foreign_key ) );
}

char *strdup( where );

/* Usage */
boolean query_not_null_drop_down_boolean(
    DICTIONARY *dictionary,
    LIST *foreign_key_list,
    int highest_index );

/* Process */
for (   int index = 0;
        index <= highest_index;
        index++ )
{
    char *query_drop_down_key(
        foreign_key_list,
        index,
        MULTI_ATTRIBUTE_KEY_DELIMITER );

    char *dictionary_get(
        query_drop_down_key(),
        dictionary );

    if ( string_strcmp(
        dictionary_get(),
        QUERY_NOT_NULL ) == 0 )
    {
        return 1;
    }
}
return 0;

/* Attribute */
LIST *foreign_key_list;
char *where;
```

QUERY_ATTRIBUTE_LIST

```
/* Usage */
QUERY_ATTRIBUTE_LIST *query_attribute_list_new(
    LIST *folder_attribute_append_isa_list,
    DICTIONARY *dictionary,
    int relation_mto1_isa_list_length,
    char *attribute_not_null );

/* Process */
QUERY_ATTRIBUTE_LIST *
    query_attribute_list_calloc(
        void );

LIST *list = list_new();

FOLDER_ATTRIBUTE *folder_attribute =
    list_first(
        folder_attribute_append_isa_list );

char *query_attribute_list_table_name(
    int relation_mto1_isa_list_length,
    char *attribute_not_null,
    char *folder_attribute->folder_name );

LIST *query_date_time_between_list(
    folder_attribute_append_isa_list,
    dictionary,
    query_attribute_list_table_name() );

for FOLDER_ATTRIBUTE *folder_attribute in
    folder_attribute_append_isa_list
{
    if ( query_date_time_between_exists(
        folder_attribute->attribute_name,
        query_date_time_between_list() )
    {
        continue;
    }

    char *query_attribute_list_table_name(
        relation_mto1_isa_list_length,
        attribute_not_null,
        folder_attribute->folder_name );

    QUERY_ATTRIBUTE *query_attribute =
        query_attribute_new(
            dictionary,
            query_attribute_list_table_name(),
            folder_attribute->attribute_name,
            folder_attribute->
                attribute->
                datatype_name ) );

    if ( query_attribute )
    {
        list_set(
            list,
            query_attribute );
    }
}

/* Note: it frees char *where in each list */
char *query_attribute_list_where(
    LIST *query_date_time_between_list,
    LIST *list /* query_attribute_list */ );

/* Usage */
LIST *query_attribute_list_name_list(
    LIST *query_attribute_list->
        query_date_time_between_list,
    LIST *query_attribute_list->list
        /* query_attribute_list */ );

/* Process */

/* Attributes */
LIST *query_date_time_between_list;
LIST *list;
char *where;
```

QUERY_DATE_TIME_BETWEEN (1)

```
/* Usage */
LIST *query_date_time_between_list(
    LIST *folder_attribute_append_isa_list,
    DICTIONARY *dictionary,
    char *table_name );

/* Process */
LIST *attribute_trimmed_list = list_new();
LIST *list = list_new();

for FOLDER_ATTRIBUTE *folder_attribute in
    folder_attribute_append_isa_list
{
    QUERY_RELATION *query_relation =
        query_relation_new(
            dictionary,
            folder_attribute->attribute_name,
            folder_attribute->
                attribute->
                datatype_name );

    if ( !query_relation
    ||   query_relation->query_relation != between )
    {
        continue;
    }

    boolean attribute_is_date(
        folder_attribute->
            attribute->
            datatype_name );

    if ( attribute_is_date() )
    {
        list_set(
            attribute_trimmed_list,
            attribute_name_trim_datatype(
                folder_attribute->attribute_name,
                1 /* attribute_is_date */,
                0 /* not attribute_is_time */ ) );

        continue;
    }

    boolean attribute_is_time(
        folder_attribute->
            attribute->
            datatype_name );

    if ( !attribute_is_time() )
    {
        continue;
    }

    char *query_date_time_between_date_attribute_name(
        attribute_trimmed_list,
        folder_attribute->attribute_name
            /* time_attribute_name */ );

    if ( query_date_time_between_date_attribute_name() ) )
    {
        list_set(
            list,
            query_date_time_between_new(
                dictionary,
                table_name,
                query_date_time_between_date_attribute_name(),
                folder_attribute->attribute_name
                    /* time_attribute_name */ ) );
    }
}

/* Usage */
char *query_date_time_between_date_attribute_name(
        LIST *attribute_trimmed_list,
        char *time_attribute_name );

/* Process */
char *attribute_name_trim_datatype(
    time_attribute_name,
    0 /* not attribute_is_date */,
    1 /* attribute_is_time */ );

if ( !attribute_name_trim_datatype() ) return (char *)0;

if ( list_string_exists(
        attribute_name_trim_datatype(),
        attribute_trimmed_list ) )
{
    sprintf(date_attribute_name[],
            "%s_date",
            attribute_name_trim_datatype() );

    return strdup( date_attribute_name );
}
```

QUERY_DATE_TIME_BETWEEN (2)

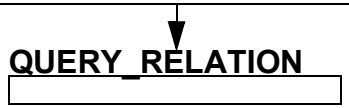
```
/* Usage */
QUERY_DATE_TIME_BETWEEN *
   query_date_time_between_new(
      DICTIONARY *dictionary,
      char *table_name,
      char *date_attribute_name,
      char *time_attribute_name );

/* Process */
QUERY_DATE_TIME_BETWEEN *
   query_date_time_between_calloc(
      void );

char *query_date_time_between_date_from_data(
   DICTIONARY *dictionary,
   char *date_attribute_name,
   char *PROMPT_LOOKUP_FROM_PREFIX );

if ( !query_date_time_between_date_from_data() )
{
   return (QUERY_DATE_TIME_BETWEEN *)0;
}

char *query_date_time_between_date_to_data(
   DICTIONARY *dictionary,
   char *date_attribute_name,
   char *PROMPT_LOOKUP_TO_PREFIX );

if ( !query_date_time_between_date_to_data() )
{
   return (QUERY_DATE_TIME_BETWEEN *)0;
}

char *query_date_time_between_time_from_data(
   DICTIONARY *dictionary,
   char *time_attribute_name,
   char *PROMPT_LOOKUP_FROM_PREFIX );

if ( !query_date_time_between_time_from_data() )
{
   return (QUERY_DATE_TIME_BETWEEN *)0;
}

char *query_date_time_between_time_to_data(
   DICTIONARY *dictionary,
   char *time_attribute_name,
   char *PROMPT_LOOKUP_TO_PREFIX );

if ( !query_date_time_between_time_to_data() )
{
   return (QUERY_DATE_TIME_BETWEEN *)0;
}

char *query_date_time_between_where(
   char *table_name,
   char *date_attribute_name,
   char *time_attribute_name,
   char *date_from_data,
   char *date_to_data,
   char *time_from_data,
   char *time_to_data );

/* Usage */
boolean query_date_time_between_exists(
   char *attribute_name,
   LIST *query_date_time_between_list );

/* Process */
for QUERY_DATE_TIME_BETWEEN *
   query_date_time_between in
      query_date_time_between_list
{
   if (strcmp(
         query_date_time_between->
            date_attribute_name,
         attribute_name ) == 0
   ||  strcmp(
         query_date_time_between->
            time_attribute_name,
         attribute_name ) == 0 )
   {
      return 1;
   }
}
return 0;

/* Usage */
LIST *query_date_time_between_attribute_name_list(
   LIST *query_date_time_between_list );

/* Process */
LIST *attribute_name_list = list_new();

for QUERY_DATE_TIME_BETWEEN *
   query_date_time_between in
      query_date_time_between_list
{
   list_set(
      attribute_name_list,
      query_date_time_between->
         date_attribute_name );

   list_set(
      attribute_name_list,
      query_date_time_between->
         time_attribute_name_list );
}

/* Attributes */
char *date_attribute_name;
char *date_from_data;
char *date_to_data;
char *time_attribute_name;
char *time_from_data;
char *time_to_data;
char *where;
```

QUERY_ATTRIBUTE (1)

```
/* Usage */
QUERY_ATTRIBUTE *query_attribute_new(
    DICTIONARY *dictionary,
    char *table_name,
    char *attribute_name,
    char *attribute->datatype_name );

/* Process */
QUERY_ATTRIBUTE *query_attribute_calloc(
    void );

boolean attribute_is_yes_no(
    attribute_name );

if ( attribute_is_yes_no() )
{
    char *where =
        char *query_attribute_yes_no_where(
            dictionary,
            table_name,
            attribute_name );

    if ( !where )
        return NULL;
    else
        return this;
}

QUERY_RELATION *query_relation =
    query_relation_new(
        dictionary,
        attribute_name,
        datatype_name );

if ( !query_relation ) return NULL;

boolean attribute_is_number(
    datatype_name );

boolean attribute_is_date_time(
    datatype_name );

boolean attribute_is_current_date_time(
    datatype_name );

if (query_relation->query_relation_enum == is_null )
{
    char *where =
        query_attribute_is_null_where(
            char *attribute_full_attribute_name(
                table_name,
                attribute_name ) );
}
else
if (query_relation->query_relation_enum == not_null )
{
    char *where =
        query_attribute_not_null_where(
            char *attribute_full_attribute_name(
                table_name,
                attribute_name ) );
}
else
if (query_relation->query_relation_enum == query_or )
{
    char *where =
        char *query_attribute_or_where(
            dictionary,
            table_name,
            attribute_name,
            attribute_is_number() );
}
else
if (query_relation->query_relation_enum ==
        query_between )
{
    char *where =
        char *query_attribute_between_where(
            dictionary,
            table_name,
            attribute_name,
            attribute_is_number(),
            attribute_is_date_time(),
            attribute_is_current_date_time() );
}
else
if ( query_relation->query_relation_enum ==
        query_begins )
{
    char *where =
        char *query_attribute_begins_where(
            DICTIONARY *dictionary,
            char *table_name,
            char *attribute_name );
}
else
if (query_relation->query_relation_enum ==
        query_contains )
{
    char *where =
        char *query_attribute_seems_where(
            dictionary,
            table_name,
            attribute_name );
}
else
if (query_relation->query_relation_enum ==
        query_not_contains )
{
    char *where =
        char *query_attribute_not_contains_where(
            dictionary,
            table_name,
            attribute_name );
}
else
if (query_relation->query_relation_enum ==
        query_equal )
{
    char *where =
        char *query_attribute_equal_where(
            DICTIONARY *dictionary,
            char *table_name,
            char *attribute_name,
            boolean attribute_is_number() );
}
else
if (query_relation->query_relation_enum ==
        query_not_equal )
{
    char *where =
        char *query_attribute_not_equal_where(
            DICTIONARY *dictionary,
            char *table_name,
            char *attribute_name,
            boolean attribute_is_number() );
}
else
```

QUERY_ATTRIBUTE (2)

```
/* Process (continued) */
{
    char *where =
        char *query_attribute_generic_where(
            DICTIONARY *dictionary,
            char *table_name,
            char *attribute_name,
            char *query_relation->character_string,
            boolean attribute_is_number() );
}

if ( !where ) this = NULL;

/* Attributes */
char *attribute_name;
QUERY_RELATION *query_relation;
boolean attribute_is_yes_no;
boolean attribute_is_number;
boolean attribute_is_date_time;
boolean attribute_is_current_date_time;
char *where;
```



QUERY_ATTRIBUTE (3)

```
/* Usage */
char *query_attribute_yes_no_where(
    DICTIONARY *dictionary,
    char *table_name,
    char *attribute_name );

/* Process */
char *dictionary_get(
    attribute_name,
    dictionary );

if ( !dictionary_get() ) return (char *)0;

char *attribute_full_attribute_name(
    table_name,
    attribute_name );

if ( *dictionary_get() == 'y' )
{
    sprintf(    where[],
                "%s = 'y'",
                attribute_full_attribute_name() );
}
else
{
    sprintf(    where[],
                "(%s = 'n' or %s is null)",
                attribute_full_attribute_name(),
                attribute_full_attribute_name() );
}
```

```
/* Usage */
char *query_attribute_or_where(
    DICTIONARY *dictionary,
    char *table_name,
    char *attribute_name,
    boolean attribute_is_number() );

/* Process */
boolean first_time = 1;

char *attribute_full_attribute_name(
    table_name,
    attribute_name );

char *query_dictionary_extract(
    dictionary,
    attribute_name,
    PROMPT_LOOKUP_FROM_PREFIX,
    (char *)0 /* PROMPT_LOOKUP_TO_PREFIX */,
    attribute_is_number );

if ( !query_dictionary_extract() ) return (char *)0;

char *ptr = char where[];
char string_delimiter( query_dictionary_extract() );
ptr += sprintf( ptr, "(" );

for char piece_buffer[] in
    piece(
        string_delimiter(),
        query_dictionary_extract() )
{
    if ( first_time )
        first_time = 0;
    else
        ptr += sprintf( ptr, " or " );

    if ( attribute_is_number )
    {
        snprintf(   expression[],
                    sizeof ( expression ),
                    "%s = %s",
                    attribute_full_attribute_name(),
                    piece_buffer );
    }
    else
    {
        snprintf(   expression[],
                    sizeof ( expression ),
                    "%s = '%s'",
                    attribute_full_attribute_name(),
                    piece_buffer );
    }

    ptr += sprintf( ptr, "%s", expression  );
}

ptr += sprintf( ptr, ")" );
```

QUERY_ATTRIBUTE (4)

```
/* Usage */
char *query_attribute_between_where(
    DICTIONARY *dictionary,
    char *table_name,
    char *attribute_name,
    boolean attribute_is_number(),
    boolean attribute_is_date_time(),
    boolean attribute_is_current_date_time() );

/* Process */
char *attribute_full_attribute_name(
    table_name,
    attribute_name );

char *query_attribute_from_data(
    dictionary,
    attribute_name,
    attribute_is_number );

if ( !query_attribute_from_data() ) return (char *)0;

char *query_attribute_to_data(
    dictionary,
    attribute_name,
    attribute_is_number,
    attribute_is_date_time,
    attribute_is_current_date_time );

if ( !query_attribute_to_data() ) return (char *)0;

if ( attribute_is_number )
{
    snprintf(
        where[],
        sizeof ( where ),
        "%s between %s and %s",
        attribute_full_attribute_name(),
        from_data,
        query_attribute_to_data() );
}
else
{
    snprintf(
        where[],
        sizeof ( where ),
        "%s between '%s' and '%s'",
        attribute_full_attribute_name(),
        query_attribute_to_data() );
}

/*  Usage */
char *query_attribute_seems_where(
    DICTIONARY *dictionary,
    char *table_name,
    char *attribute_name );

/* Process */
char *query_dictionary_extract(
    dictionary,
    attribute_name,
    PROMPT_LOOKUP_FROM_PREFIX,
    (char *)0 /* PROMPT_LOOKUP_TO_PREFIX */,
    0 /* not attribute_is_number */ );

if ( !query_dictionary_extract() ) return null;

char *attribute_full_attribute_name(
    table_name,
    attribute_name );

char *query_attribute_like_where(
    query_dictionary_extract() );

snprintf(
    where[],
    sizeof ( where ),
    "(%s %s or %s sounds like '%')",
    attribute_full_attribute_name(),
    query_attribute_like_where(),
    attribute_full_attribute_name(),
    query_dictionary_extract() );

free( query_attribute_like_where() );

char *strdup( where );

/*  Usage */
char *query_attribute_not_contains_where(
    DICTIONARY *dictionary,
    char *table_name,
    char *attribute_name );

/* Process */
char *query_dictionary_extract(
    dictionary,
    attribute_name,
    PROMPT_LOOKUP_FROM_PREFIX,
    (char *)0 /* PROMPT_LOOKUP_TO_PREFIX */,
    0 /* not attribute_is_number */ );

if ( !query_dictionary_extract() ) return null;

char *attribute_full_attribute_name(
    table_name,
    attribute_name );

snprintf(
    where[],
    sizeof ( where ),
    "%s not like '%c%s%c'",
    attribute_full_attribute_name(),
    '%',
    query_dictionary_extract() );
    '%' );

char *strdup( where );

/* Usage */
char *query_attribute_like_where(
    char *query_dictionary_extract );

/* Process */
string_strcpy(
    source_destination[],
    query_dictionary_extract );

char *string_search_replace_character(
    source_destination,
    ' ' /* search_character */,
    '%' /* destination_character */ );

snprintf(
    where[],
    sizeof ( where ),
    "like '%c%s%c'",
    '%',
    string_search_replace_character(),
    '%' );

char *strdup( where );
```

QUERY_ATTRIBUTE (5)

```
/* Usage */
char *query_attribute_from_data(
    DICTIONARY *dictionary,
    char *attribute_name,
    boolean attribute_is_number() );

/* Process */
char *query_dictionary_extract(
    dictionary,
    attribute_name,
    PROMPT_LOOKUP_FROM_PREFIX,
    (char *)0 /* PROMPT_LOOKUP_TO_PREFIX */,
    attribute_is_number );

if ( !query_dictionary_extract() ) return (char *)0;

/* Usage */
char *query_attribute_to_data(
    DICTIONARY *dictionary,
    char *attribute_name,
    boolean attribute_is_number()
    boolean attribute_is_date_time(),
    boolean attribute_is_current_date_time() );

/* Process */
char *query_dictionary_extract(
    dictionary,
    attribute_name,
    (char *)0 /* PROMPT_LOOKUP_FROM_PREFIX */,
    PROMPT_LOOKUP_TO_PREFIX,
    attribute_is_number );

if ( !query_dictionary_extract() ) return (char *)0;

if ( (  attribute_is_date_time
||      attribute_is_current_date_time )
&&   !string_character_exists(
            query_dictionary_extract(),
            ' ' ) )
{
    sprintf(
            to_data[],
            "%s 23:23:59",
            query_dictionary_extract() );

    return strdup( to_data );
}
else
{
    return query_dictionary_extract();
}
```

QUERY_ISA

```
/* Usage */
QUERY_ISA *query_isa_new(
    char *application_name,
    LIST *relation_mto1_isa_list );

/* Process */
QUERY_ISA *query_isa_calloc(
    void );

if ( !list_length( relation_mto1_isa_list ) ) return this;

char *query_isa_join_where(
    application_name,
    relation_mto1_isa_list );

/* Usage */
char *query_isa_join_where(
    char *application_name,
    LIST *relation_mto1_isa_list );

/* Process */
char join_where[];
char *ptr = join_where;

for RELATION_MTO1 *relation_mto1 in
    relation_mto1_isa_list
{
    if ( ptr != where ) ptr += sprintf( ptr, " and " );

    ptr +=
        sprintf(
            ptr,
            "%s",
            query_isa_where_join(
                application_name,
                relation_mto1->many_folder_name,
                relation_mto1->
                    relation_foreign_key_list,
                relation_mto1->
                    one_folder_name,
                relation_mto1->
                    one_folder->
                    folder_attribute_primary_key_list
                    /* one_folder_primary_key_list */ ) );
}

/* Usage */
char *query_isa_where_join(
    char *application_name,
    char *many_folder_name,
    LIST *relation_foreign_key_list,
    char *one_folder_name,
    LIST *one_folder_primary_key_list );

/* Process */
static char join[];
char *ptr = join;

char *many_table_name =
    strdup(
        appaserver_table_name(
            many_folder_name ) );

char *one_table_name =
    appaserver_table_name(
        one_folder_name );

list_rewind( relation_foreign_key_list );

for char *primary_key in one_folder_primary_key_list
{
    if ( ptr != join ) ptr += sprintf( ptr, " and " );

    ptr +=
        sprintf(
            ptr,
            "%s.%s = %s.%s",
            many_table_name,
            (char *)list_get( relation_foreign_key_list ),
            one_table_name,
            primary_key );

    list_next( relation_foreign_key_list );
}

free( many_table_name );

/* Attribute */
char *join_where;
```

QUERY_SELECT (1)

```
/* Usage */
LIST *query_select_table_edit_list(
    char *application_name,
    char *folder_name,
    LIST *folder_attribute_append_isa_list,
    LIST *no_display_name_list,
    LIST *exclude_lookup_attribute_name_list,
    int relation_mto1_isa_list_length,
    char *attribute_not_null );

/* Process */
LIST *query_select_list = list_new();

for FOLDER_ATTRIBUTE *folder_attribute in
    folder_attribute_append_isa_list
{
    boolean query_select_skip_boolean(
        folder_name,
        folder_attribute->folder_name,
        folder_attribute->attribute_name,
        no_display_name_list,
        exclude_lookup_attribute_name_list,
        folder_attribute->primary_key_index,
        attribute_not_null );

    if ( query_select_skip_boolean() ) continue;

    list_set(
        query_select_list,
        QUERY_SELECT *query_select_new(
            relation_mto1_isa_list_length,
            attribute_not_null,
            folder_attribute->folder_name,
            folder_attribute->attribute_name ) );
}

QUERY_SELECT *query_select_attribute_not_null(
    folder_attribute_append_isa_list,
    attribute_not_null );

if ( query_select_attribute_not_null() )
{
    list_set(
        query_select_list,
        query_select_attribute_not_null() );
}

/* Usage */
LIST *query_select_primary_list(
    char *application_name,
    char *folder_name,
    LIST *folder_attribute_append_isa_list,
    char *additional_attribute_name,
    int relation_mto1_isa_list_length );

/* Process */
LIST *query_select_list = list_new();

for FOLDER_ATTRIBUTE *folder_attribute in
    folder_attribute_append_isa_list
{
    if (    strcmp(
                folder_attribute->folder_name,
                folder_name ) != 0
    ||      !folder_attribute->primary_key_index )
    {
        continue;
    }

    list_set(
        query_select_list,
        QUERY_SELECT *query_select_new(
            relation_mto_isa_list_length,
            (char *)0 /* attribute_not_null */,
            folder_attribute->folder_name,
            folder_attribute->attribute_name ) );
}

if ( additional_attribute_name )
{
    list_set(
        query_select_list,
        QUERY_SELECT *query_select_new(
            0 /* relation_mto1_isa_list_length */,
            (char *)0 /* attribute_not_null */,
            (char *)0 /* folder_name */,
            additional_attribute_name ) );
}

/* Usage */
QUERY_SELECT *query_select_new(
    int relation_mto1_isa_list_length,
    char *attribute_not_null,
    char *folder_name,
    char *attribute_name );

/* Process */
QUERY_SELECT *query_select_calloc(
    void );

char *query_select_full_attribute_name(
    relation_mto1_isa_list_length,
    attribute_not_null,
    folder_name,
    attribute_name );

/* Attributes */
char *attribute_name;
char *full_attribute_name;
```

QUERY_SELECT (2)

```
/* Usage */
char *query_select_full_attribute_name(
    int relation_mto1_isa_list_length,
    char *attribute_not_null,
    char *folder_name,
    char *attribute_name );

/* Process */
char *query_table_name(
    folder_name,
    relation_mto1_isa_list_length,
    attribute_not_null );

char *attribute_full_attribute_name(
    query_table_name(),
    attribute_name );

/* Usage */
LIST *query_select_list(
    LIST *folder_attribute_list );

/* Process */
LIST *list = list_new();

for FOLDER_ATTRIBUTE *folder_attribute in
    folder_attribute_list
{
    list_set(
        list,
        query_select_new(
            0 /* relation_mto1_isa_list_length */,
            (char *)0 /* attribute_not_null */,
            folder_attribute->attribute_name ) );
}
```

```
/* Usage */
boolean query_select_skip_boolean(
    char *folder_name,
    char *folder_attribute_folder_name,
    char *folder_attribute->attribute_name,
    LIST *no_display_name_list,
    LIST *exclude_lookup_attribute_name_list,
    int folder_attribute->primary_key_index,
    char *attribute_not_null );

/* Process */
if (    attribute_not_null
&&  strcmp(
            attribute_name,
            attribute_not_null ) == 0 )
{
    return 0;
}

if (    list_string_exists(
            attribute_name,
            no_display_name_list )
&&  !primary_key_index )
{
    return 1;
}

if (    list_string_exists(
            attribute_name,
            exclude_lookup_attribute_name_list )
&&  !primary_key_index )
{
    return 1;
}

if (    strcmp(
            folder_attribute_folder_name,
            folder_name ) != 0
&&  primary_key_index )
{
    return 1;
}

return 0;

/* Usage */
QUERY_SELECT *query_select_attribute_not_null(
    LIST *folder_attribute_append_isa_list,
    char *attribute_not_null );

/* Process */
if ( !attribute_not_null ) return null;

FOLDER_ATTRIBUTE *folder_attribute_seek(
    (char *)0 /* folder_name */,
    attribute_not_null /* attribute_name */,
    folder_attribute_append_isa_list );

if ( !folder_attribute_seek() )
{
    QUERY_SELECT *query_select_new(
        0 /* relation_mto1_is_list_length */,
        (char *)0 /* attribute_not_null */,
        (char *)0 /* folder_name */,
        attribute_not_null );
}
```

QUERY_SELECT (3)

```
/* Usage */
char *query_select_list_string(
    LIST *query_select_list,
    LIST *common_name_list );

/* Process */
char list_string[];
char *ptr = list_string;

for QUERY_SELECT *query_select in query_select_list
{
    if ( ptr != list_string ) ptr += sprintf( ptr, "," );

    if (    list_length( common_name_list )
    &&   list_last_boolean( query_select_list ) )
    {
        ptr += sprintf( ptr,
            "concat(%s,' ['",
            query_select->full_attribute_name );
    }
    else
    {
        ptr += sprintf( ptr,
            "%s",
            query_select->full_attribute_name );
    }
}

for char *common_name in common_name_list
{
    ptr += sprintf( ptr,
        ",%s",
        common_name );

    if (    list_length( common_name_list ) > 1
    &&   !list_last_boolean( common_name_list ) )
    {
        ptr += sprintf( ptr, ",'/'" );
    }

    if ( list_last_boolean( common_name_list ) )
    {
        ptr += sprintf( ptr, ",']')" );
    }
}

/* Usage */
LIST *query_select_name_list(
    LIST *query_select_list );

/* Process */
LIST *name_list = list_new();

for QUERY_SELECT *query_select in query_select_list
{
    list_set( name_list, query_select->attribute_name );
}
```

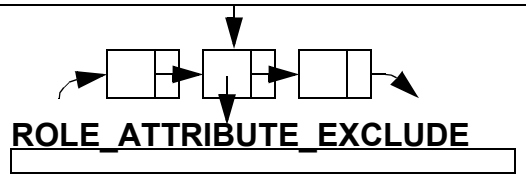
QUERY_RELATION

```
/* Usage */
QUERY_RELATION *query_relation_new(
    DICTIONARY *dictionary
        /* query_dictionary|drillthru_dictionary */,
    char *attribute_name,
    char *datatype_name );

/* Process */
QUERY_RELATION *query_relation_calloc( void );

char *query_relation_key(
    char *QUERY_RELATION_OPERATOR_PREFIX,
    char *attribute_name );

char *query_relation_yes_no_key(
    char *attribute_name );

char *query_relation_operator_extract(
    DICTIONARY *dictionary,
    char *query_relation_key(),
    char *query_relation_yes_no_key() );

if ( !query_relation_operator_extract() ) return NULL;

enum query_relation query_relation_enum(
    char *QUERY_EQUAL,
    char *QUERY_NOT_EQUAL,
    char *QUERY_LESS_THAN,
    char *QUERY_LESS_THAN_EQUAL_TO,
    char *QUERY_GREATER_THAN,
    char *QUERY_GREATER_THAN_EQUAL_TO,
    char *QUERY_BETWEEN,
    char *QUERY_BEGINS,
    char *QUERY_CONTAINS,
    char *QUERY_NOT_CONTAINS,
    char *QUERY_OR,
    char *QUERY_NULL,
    char *QUERY_NOT_NULL,
    char *datatype_name,
    query_relation_operator_extract() );

char *query_relation_character_string(
    enum query_relation_enum() );

/* Attributes */
enum query_relation query_relation_enum;
char *character_string;
```

```
enum query_relation {
    query_equal
    query_not_equal
    query_less_than
    query_less_than_equal_to
    query_greater_than
    query_greater_than_equal_to
    query_between
    query_begins
    query_contains
    query_not_contains
    query_or
    query_is_null
    query_not_null
```

ROLE

```
/* Usage */
ROLE *role_fetch(
    char *role_name,
    boolean fetch_role_attribute_exclude_list );

/* Process */
boolean grace_no_cycle_colors_exists =
    folder_attribute_exists(
        ROLE_TABLE,
        “grace_no_cycle_colors_yn” );

char *role_select(
    boolean grace_no_cycle_colors_exists );

char *role_primary_where(
    char *role_name );

char *appaserver_system_string(
    char *role_select(),
    char *ROLE_TABLE,
    char *role_primary_where() );

char *string_pipe_input(
    appaserver_system_string() );

ROLE *role_parse(
    role_name,
    fetch_role_attribute_exclude_list,
    string_pipe_input() );

/* Usage */
ROLE *role_parse(
    char *role_name,
    boolean fetch_role_attribute_exclude_list,
    char *input );

ROLE *role_calloc( void );

if ( fetch_role_attribute_exclude_list )
{
    LIST *role_attribute_exclude_list( role_name );
}

/* Usage */
LIST *role_name_list(
    const char *ROLE_TABLE );

/* Process */
sprintf(
    system_string[],
    “select.sh role %s”,
    ROLE_TABLE );

LIST *list_pipe( system_string );

/* Attributes */
char *role_name;
boolean grace_no_cycle_colors;
boolean folder_count;
boolean override_row_restrictions;
LIST *role_attribute_exclude_list;
```



ROLE_ATTRIBUTE_EXCLUDE

```
/* Usage */
LIST *role_attribute_exclude_list(
    char *role_name );

/* Process */
char *role_primary_where( role_name );

char *appaserver_system_string(
    ROLE_ATTRIBUTE_EXCLUDE_SELECT,
    ROLE_ATTRIBUTE_EXCLUDE_TABLE,
    role_primary_where() /* where */ );

LIST *role_attribute_exclude_system_list(
    appaserver_system_string() );

/* Usage */
LIST *role_attribute_exclude_system_list(
    char *appaserver_system_string() );

/* Process */
FILE *appaserver_input_pipe(
    appaserver_system_string() );

LIST *list = list_new();

while ( string_input( input[], appaserver_input_pipe() )
{
    list_set(
        list,
        ROLE_ATTRIBUTE_EXCLUDE *
            role_attribute_exclude_parse(
                input ) );
}

void pclose( appaserver_input_pipe() );

/* Usage */
ROLE_ATTRIBUTE_EXCLUDE *
    role_attribute_exclude_parse(
        char *input );

/* Process */
ROLE_ATTRIBUTE_EXCLUDE *
    role_attribute_exclude_calloc(
        void );

/* Usage */
LIST *role_attribute_exclude_lookup_name_list(
    const char *APPASERVER_LOOKUP_STATE,
    const char *APPASERVER_UPDATE_STATE,
    LIST *role_attribute_exclude_list() );

/* Process */
LIST *list = list_new();

list_set_list(
    list,
    role_attribute_exclude_name_list(
        APPASERVER_LOOKUP_STATE
            /* role_permission */,
        role_attribute_exclude_list ) );

list_set_list(
    list,
    role_attribute_exclude_name_list(
        APPASERVER_UPDATE_STATE
            /* role_permission */,
        role_attribute_exclude_list ) );

/* Usage */
LIST *role_attribute_exclude_insert_name_list(
    const char *APPASERVER_INSERT_STATE,
    LIST *role_attribute_exclude_list() );

/* Process */
LIST *role_attribute_exclude_name_list(
    APPASERVER_INSERT_STATE
        /* role_permission */,
    role_attribute_exclude_list() );

/* Usage */
LIST *role_attribute_exclude_name_list(
    char *role_permission,
    LIST *role_attribute_exclude_list() );

/* Process */

/* Usage */
boolean role_attribute_exclude_exists_permission(
    char *role_permission,
    char *attribute_name,
    LIST *role_attribute_exclude_list() );

/* Process */

/* Attributes */
char *attribute_name;
char *permission;
```

ROLE_APPASERVER_USER

```
/* Usage */
LIST *role_appaserver_user_name_list(
    const char *ROLE_APPASERVER_USER_TABLE,
    char *full_name,
    char *contact_key,
    boolean entity_contact_key_boolean );

/* Process */

char *entity_primary_where(
    APPASERVER_FULL_NAME_COLUMN,
    APPASERVER_CONTACT_KEY_COLUMN,
    full_name,
    contact_key,
    entity_contact_key_boolean );

char *appaserver_system_string(
    ROLE_NAME_COLUMN /* select */,
    ROLE_APPASERVER_USER_TABLE,
    entity_primary_where() );

LIST *list_system_string_fetch( appaserver_system_string() );

/* Attributes */
```

LOGIN_DEFAULT_ROLE

```
/* Usage */
char *login_default_role_name(
    const char *LOGIN_DEFAULT_ROLE_TABLE,
    char *full_name,
    char *contact_key,
    boolean entity_contact_key_boolean );

/* Process */

char *entity_primary_where(
    APPASERVER_FULL_NAME_COLUMN,
    APPASERVER_CONTACT_KEY_COLUMN,
    full_name,
    contact_key,
    entity_contact_key_boolean );

char *appaserver_system_string(
    ROLE_NAME_COLUMN /* select */,
    LOGIN_DEFAULT_ROLE_TABLE,
    char *entity_primary_where() );

char *string_system_input(
    appaserver_system_string() );

/* Attributes */
```

ROLE_FOLDER (1)

```
/* Usage */
LIST *role_folder_list(
    char *role_name,
    char *folder_name );

/* Process */
char *role_folder_where(
    const char *ROLE_FOLDER_TABLE,
    const char *FOLDER_PRIMARY_KEY,
    char *role_name,
    char *folder_name );

char *role_folder_left_join(
    ROLE_FOLDER_TABLE,
    FOLDER_TABLE,
    FOLDER_PRIMARY_KEY );

char *role_folder_system_string(
    ROLE_FOLDER_SELECT,
    ROLE_FOLDER_TABLE,
    role_folder_where(),
    role_folder_left_join() );

LIST *role_folder_system_list(
    role_folder_system_string() );

/* Usage */
LIST *role_folder_lookup_list(
    char *role_name );

/* Process */
char *role_folder_lookup_where(
    const char *APPASERVER_LOOKUP_STATE,
    const char *APPASERVER_UPDATE_STATE,
    char *role_name );

char *role_folder_left_join(
    ROLE_FOLDER_TABLE,
    FOLDER_TABLE,
    FOLDER_PRIMARY_KEY );

char *role_folder_system_string(
    ROLE_FOLDER_SELECT,
    ROLE_FOLDER_TABLE,
    role_folder_lookup_where(),
    role_folder_left_join() );

LIST *role_folder_system_list(
    role_folder_system_string() );

/* Usage */
LIST *role_folder_insert_list(
    char *role_name );

/* Process */
char *role_folder_insert_where(
    const char *APPASERVER_INSERT_STATE,
    char *role_name );

char *role_folder_left_join(
    ROLE_FOLDER_TABLE,
    FOLDER_TABLE,
    FOLDER_PRIMARY_KEY );

char *role_folder_system_string(
    ROLE_FOLDER_SELECT,
    ROLE_FOLDER_TABLE,
    role_folder_insert_where(),
    role_folder_left_join() );

LIST *role_folder_system_list(
    role_folder_system_string() );

/* Usage */
LIST *role_folder_system_list(
    char *role_folder_system_string() );

/* Process */
FILE *role_folder_input_pipe(
    char *role_folder_system_string );

for char input[] in
    string_input( role_folder_input_pipe() )
{
    list_set( list, role_folder_parse( input ) );
}

void pclose( role_folder_input_pipe() );

/* Usage */
ROLE_FOLDER *role_folder_parse(
    char *input );

/* Process */
if ( strcasecmp( folder_name, “null” ) == 0 )
    return NULL;

ROLE_FOLDER *role_folder_new(
    role_name,
    folder_name,
    permission );

/* Usage */
ROLE_FOLDER *role_folder_new(
    char *role_name,
    char *folder_name,
    char *permission );

/* Process */
ROLE_FOLDER *role_folder_calloc(
    void );

/* Attributes */
char *role_name;
char *folder_name;
char *permission;
char *subschema_name;
```

ROLE_FOLDER (2)

```
/* Usage */
char *role_folder_left_join(
    const char *ROLE_FOLDER_TABLE,
    const char *FOLDER_TABLE,
    const char *FOLDER_PRIMARY_KEY );

/* Process */
if ( *left_join ) return left_join;

snprintf(
    static char left_join[],
    sizeof ( left_join ),
    "left join %s on %s.%s = %s.%s",
    FOLDER_TABLE,
    ROLE_FOLDER_TABLE,
    FOLDER_PRIMARY_KEY,
    FOLDER_TABLE,
    FOLDER_PRIMARY_KEY );

/* Usage */
char *role_folder_lookup_in_clause(
    const char *ROLE_FOLDER_TABLE,
    const char *APPASERVER_LOOKUP_STATE,
    const char *APPASERVER_UPDATE_STATE,
    const char *FOLDER_PRIMARY_KEY,
    char *role_name );

/* Process */
char *role_folder_permission_where(
    const char *APPASERVER_LOOKUP_STATE,
    const char *APPASERVER_UPDATE_STATE,
    char *role_name );

snprintf(
    static char in_clause[],
    sizeof ( in_clause ),
    "%s in ( select distinct %s from %s %s )",
    FOLDER_PRIMARY_KEY,
    FOLDER_PRIMARY_KEY,
    ROLE_FOLDER_TABLE,
    role_folder_permission_where() );

/* Usage */
LIST *role_folder_fetch_name_list(
    const char *ROLE_FOLDER_TABLE,
    const char *FOLDER_PRIMARY_KEY,
    char *role_name );

/* Process */
sprintf(
    where[],
    "role = '%s' and %s <> 'null'",
    role_name,
    FOLDER_PRIMARY_KEY );

sprintf(
    system_string[],
    "select.sh %s %s \"%s\" |"
    "sort -u",
    FOLDER_PRIMARY_KEY,
    ROLE_FOLDER_TABLE,
    where );

LIST *list_pipe( system_string );

/* Usage */
char *role_folder_system_string(
    const char *ROLE_FOLDER_SELECT,
    const char *ROLE_FOLDER_TABLE,
    char *where,
    char *role_folder_left_join() );

/* Process */
snprintf(
    system_string[],
    sizeof ( system_string ),
    "echo \"select %s from %s %s where %s;\" | sql.e",
    ROLE_FOLDER_SELECT,
    ROLE_FOLDER_TABLE,
    role_folder_left_join,
    where );

/* Usage */
LIST *role_folder_name_list(
    LIST *role_folder_list );

/* Process */

/* Usage */
ROLE_FOLDER *role_folder_permission_seek(
    char *folder_name,
    char *permission,
    LIST *role_folder_list );

/* Process */

/* Usage */
boolean role_folder_insert_exists(
    char *folder_name,
    LIST *role_folder_list );

/* Process */

/* Usage */
LIST *role_folder_subschema_folder_name_list(
    char *subschema_name,
    LIST *role_folder_list );

/* Process */

/* Usage */
LIST *role_folder_subschema_name_list(
    LIST *role_folder_lookup_list );

/* Process */

/* Usage */
LIST *role_folder_subschema_missing_folder_name_list(
    LIST *role_folder_lookup_list );

/* Process */

/* Usage */
char *role_folder_list_display(
    LIST *role_folder_list );

/* Process */
```

ROLE_FOLDER (3)

```
/* Usage */
boolean role_folder_lookup_boolean(
    const char *appaserver_lookup_state,
    const char *appaserver_update_state,
    char *folder_name,
    LIST *role_folder_list );

/* Process */
boolean role_folder_state_boolean(
    appaserver_lookup_state,
    folder_name,
    role_folder_list );

if ( role_folder_state_boolean() ) return 1;

boolean role_folder_state_boolean(
    appaserver_update_state,
    folder_name,
    role_folder_list );

* Usage */
boolean role_folder_update_boolean(
    const char *appaserver_update_state,
    char *folder_name,
    LIST *role_folder_list );

/* Process */
boolean role_folder_state_boolean(
    appaserver_update_state,
    folder_name,
    role_folder_list );

/* Usage */
boolean role_folder_insert_boolean(
    const char *appaserver_insert_state,
    char *folder_name,
    LIST *role_folder_list );

/* Process */
boolean role_folder_state_boolean(
    appaserver_insert_state,
    folder_name,
    role_folder_list );

/* Usage */
boolean role_folder_state_boolean(
    char *state,
    char *folder_name,
    LIST *role_folder_list );

/* Process */
ROLE_FOLDER *role_folder_permission_seek(
    folder_name,
    state /* permission */,
    role_folder_list );

(boolean)role_folder_permission_seek();

/* Usage */
static char *role_folder_where_string(
    const char *ROLE_FOLDER_TABLE,
    const char *FOLDER_PRIMARY_KEY,
    char *role_name_list_string );

/* Process */
```

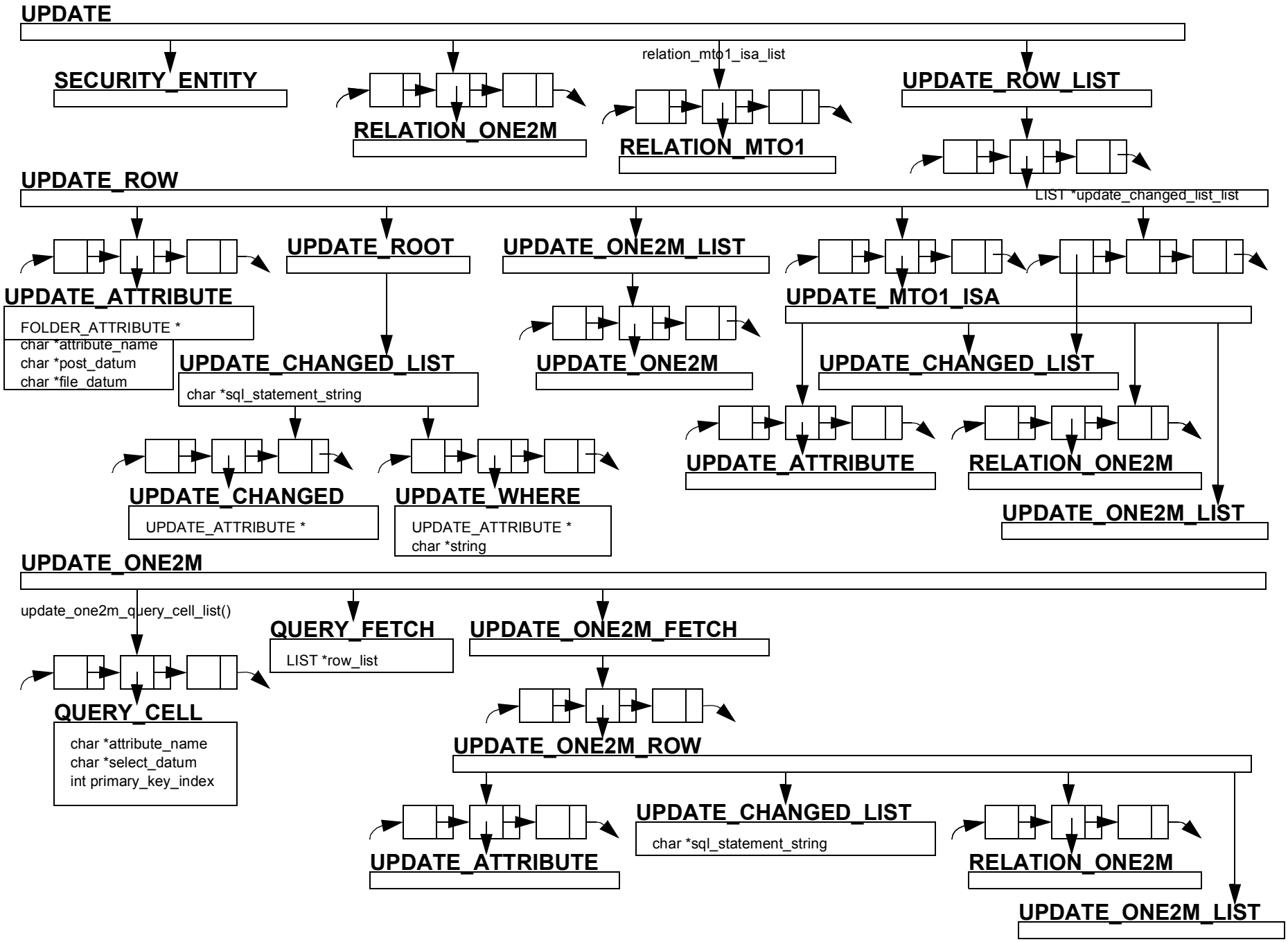
ROLE_PROCESS

```
/* Usage */
LIST *role_process_list(
    char *role_name,
    boolean fetch_process );

/* Process */
LIST *list = list_new();

char *role_primary_where( role_name );

char *role_process_system_string(
    const char *ROLE_PROCESS_SELECT,
    const char *ROLE_PROCESS_TABLE,
    char *role_primary_where()
        /* where */ );

FILE *role_process_input_pipe(
    char *role_process_system_string() );

for char input[] in role_process_input_pipe()
{
    list_set(
        list,
        role_process_parse(
            role_name,
            fetch_process,
            input ) );
}

void pclose( role_process_input_pipe() );
```

```
/* Usage */
ROLE_PROCESS *role_process_parse(
    char *role_name,
    boolean fetch_process,
    char *input );

/* Process */
ROLE_PROCESS *role_process =
    role_process_new(
        role_name,
        strdup( process_name ) );

if ( fetch_process )
{
    process =
        process_fetch(
            process_name,
            (char *)0 /* document_root_directory */,
            (char *)0 /* relative_source_directory */,
            0 /* not check_executable_inside_filesystem */,
            (char *)0 /* mount_point */ );
}

/* Usage */
ROLE_PROCESS *role_process_new(
    char *role_name,
    char *process_name );

/* Process */
ROLE_PROCESS *role_process_calloc( void );
```

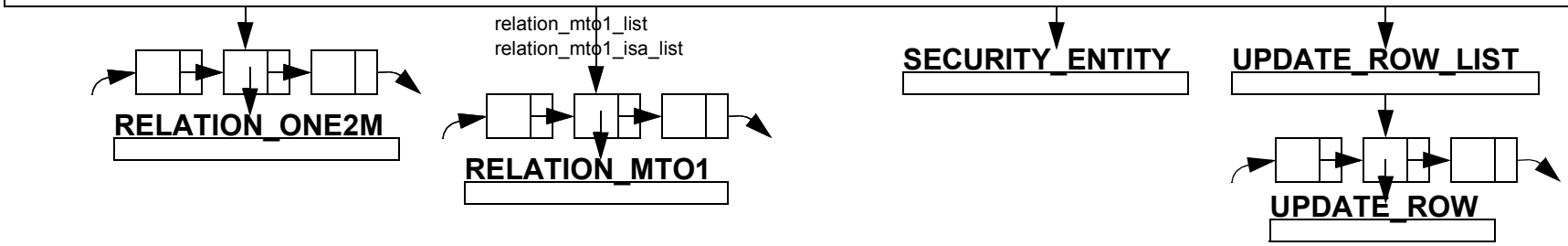
```
/* Usage */
LIST *role_process_group_name_list(
    LIST *role_process_list );

/* Process */

/* Usage */
LIST *role_process_name_list(
    char *process_group,
    LIST *role_process_list );

/* Process */

/* Usage */
LIST *role_process_missing_group_name_list(
    LIST *role_process_list );

/* Process */

/* Usage */
boolean role_process_exists(
    char *process_name,
    LIST *role_process_list );

/* Process */

/* Attributes */
char *role_name;
char *process_name;
PROCESS *process;
```

PROCESS


UPDATE
SECURITY_ENTITY
RELATION_ONE2M
relation_mto1_isa_list
RELATION_MTO1
UPDATE_ROW_LIST
LIST *update_changed_list_list
UPDATE_ROW
UPDATE_ROOT
UPDATE_ONE2M_LIST
UPDATE_ATTRIBUTE
FOLDER_ATTRIBUTE *
char *attribute_name
char *post_datum
char *file_datum
UPDATE_MTO1_ISA
UPDATE_CHANGED_LIST
char *sql_statement_string
UPDATE_ONE2M
UPDATE_CHANGED_LIST
UPDATE_ATTRIBUTE
RELATION_ONE2M
UPDATE_ONE2M_LIST
UPDATE_CHANGED
UPDATE_ATTRIBUTE *
UPDATE_WHERE
UPDATE_ATTRIBUTE *
char *string
UPDATE_ONE2M
update_one2m_query_cell_list()
QUERY_FETCH
LIST *row_list
UPDATE_ONE2M_FETCH
QUERY_CELL
char *attribute_name
char *select_datum
int primary_key_index
UPDATE_ONE2M_ROW
UPDATE_CHANGED_LIST
char *sql_statement_string
UPDATE_ATTRIBUTE
RELATION_ONE2M
UPDATE_ONE2M_LIST

UPDATE (1)

```
/* Usage */
UPDATE *update_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    boolean role->override_role_restrictions,
    DICTIONARY *dictionary_separate->
        multi_row_dictionary,
    DICTIONARY *file_dictionary,
    FOLDER_ROW_LEVEL_RESTRICTION *
        folder_row_level_restriction,
    LIST *relation_one2m_list(
        1 /* include_isa_boolean */ ),
    LIST *relation_mto1_list,
    LIST *relation_mto1_isa_list,
    LIST *folder_attribute_append_isa_list,
    PROCESS *post_change_process,
    boolean update_root_boolean );

/* Process */
UPDATE *update_calloc(
    void );

if ( folder_row_level_restriction )
{
    SECURITY_ENTITY *security_entity =
        security_entity_new(
            login_name,
            folder_name,
            folder_row_level_restriction->
                non_owner_forbid ||
            folder_row_level_restriction->
                non_owner_view_only,
            override_row_restrictions );
}

char *appaserver_error_filespecification(
    application_name );

char *appaserver_parameter_mount_point();

UPDATE_ROW_LIST *update_row_list =
    update_row_list_new(
        application_name,
        session_key,
        login_name,
        role_name,
        folder_name,
        multi_row_dictionary,
        file_dictionary,
        relation_one2m_list,
        relation_mto1_isa_list,
        folder_attribute_append_isa_list,
        post_change_process,
        update_root_boolean,
        security_entity,
        appaserver_error_filename(),
        appaserver_parameter_mount_point() );

if ( !update_row_list ) return NULL;

char *update_results_string(
    int update_row_list->cell_count,
    char *update_row_list->
        folder_name_list_string );

/* Attributes */
LIST *relation_one2m_list;
LIST *relation_mto1_list;
LIST *relation_mto1_isa_list;
SECURITY_ENTITY *security_entity;
char *appaserver_error_filename;
char *appaserver_parameter_mount_point;
UPDATE_ROW_LIST *update_row_list;
char *results_string;

/* Set externally */
char *error_string
```



UPDATE (2)

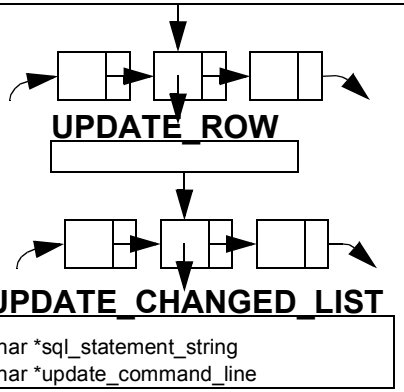
```
/* Usage */
char *update_command_line(
    const char *APPASERVER_UPDATE_STATE,
    const char *UPDATE_PREUPDATE_PREFIX,
    char *post_change_process_command_line,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *post_change_process_name,
    char *appaserver_error_filespecification,
    LIST *primary_data_list,
    LIST *update_attribute_list,
    int row_number,
    int row_count );

/* Process */
char *process_replace_command_line(
    UPDATE_PREUPDATE_PREFIX,
    (char *)0 /* application_name */,
    session_key,
    login_name,
    role_name,
    folder_name,
    (char *)0 /* target_frame */,
    APPASERVER_UPDATE_STATE,
    post_change_process_name,
    (char *)0 /* many_folder_name */,
    (char *)0 /* one_folder_name */,
    (char *)0 /* related_column */,
    (char *)0 /* update_results_string */,
    (char *)0 /* update_error_string */,
    (DICTIONARY *)0 /* operation_row_list_dictionary */,
    (DICTIONARY *)0 /* dictionary_single_row */,
    (char *)0 /* where_string */,
    row_number,
    row_count,
    0 /* parent_process_id */,
    primary_data_list,
    "y" /* execute_yn */,
    (LIST *)0 /* insert_datum_list */,
    update_attribute_list,
    (LIST *)0 /* query_cell_list */,
    appaserver_error_filespecification,
    post_change_process_command_line );
```

UPDATE (3)

```
/* Usage */
char *update_system_string(
    const char *SQL_EXECUTABLE,
    char *appaserver_error_filename );

/* Process */
snprintf(
    static system_string[],
    sizeof ( system_string ),
    "tee -a %s | %s 2>>%s",
    appaserver_error_filename,
    SQL_EXECUTABLE,
    appaserver_error_filename );

/* Usage */
char *update_sql_statement_string(
    LIST *update_column_name_list,
    LIST *update_data_list,
    LIST *primary_key_list,
    LIST *primary_data_list,
    LIST *folder_attribute_list,
    char *table_name );

/* Process */
char statement_string[];
char *ptr = statement_string;
boolean first_time = 1;

ptr += sprintf( ptr,
    "update %s set ",
    table_name );

list_rewind( update_data_list );

for char *update_column_name in
    update_column_name_list
{
    if ( first_time )
        first_time = 0;
    else
        ptr += sprintf( ptr, "," );

    FOLDER_ATTRIBUTE *folder_attribute =
        folder_attribute_seek(
            (char *)0 /* folder_name */,
            update_column_name,
            folder_attribute_list );

    if ( folder_attribute )
    {
        datatype_name =
            folder_attribute->
                attribute->
                datatype_name );
    }

    char *update_changed_set_string(
        QUERY_IS_NULL,
        update_column_name,
        datatype_name,
        list_get( update_data_list )
            /* post_datum */ );

    ptr += sprintf( ptr,
            "%s",
            update_changed_set_string() );

    list_next( update_data_list );
}

sprintf(
        ptr,
        " where %s;",
        char *folder_attribute_where_string(
            folder_attribute_list,
            primary_key_list,
            primary_data_list ) );

/* Usage */
void update_statement_execute(
    const char *SQL_EXECUTABLE,
    char *application_name,
    char *update_statement );

/* Process */
char *appaserver_error_filespecification(
    application_name );

char *update_system_string(
    SQL_EXECUTABLE,
    appaserver_error_filespecification() );

FILE *appaserver_output_pipe(
    update_system_string() );

fprintf(
    appaserver_output_pipe(),
    "%s\n",
    update_statement );

pclose( appaserver_output_pipe() );

/* Driver */
void update_stderr_display(
    LIST *update_row_list->list /* update_row_list */ );

/* Process */
for UPDATE_ROW *update_row in update_row_list
{
    for UPDATE_CHANGED_LIST *update_changed_list in
        update_row->update_changed_list_list
    {
        char *update_changed_list_display(
            update_changed_list );

        fprintf(
            stderr,
            "%s\n",
            update_changed_list_display() );
    }
}

/* Driver */
char *update->error_string =
    char *update_row_list_execute(
        SQL_EXECUTABLE,
        application_name,
        update_root_boolean,
        update->update_row_list,
        update->appaserver_error_filename );

if ( !update->error_string )
{
    void update_row_list_command_line_execute(
        update->update_row_list );
}
```

UPDATE_ROW_LIST (1)

```
/* Usage */
UPDATE_ROW_LIST *update_row_list_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    DICTIONARY *multi_row_dictionary,
    DICTIONARY *file_dictionary,
    LIST *relation_one2m_list(
        1 /* include_isa_boolean */ ),
    LIST *relation_mto1_isa_list,
    LIST *folder_attribute_append_isa_list(),
    PROCESS *post_change_process,
    boolean update_root_boolean,
    SECURITY_ENTITY *security_entity,
    char *update->
        appaserver_error_filespecification,
    char *update->
        appaserver_parameter_mount_point );

/* Process */
int dictionary_highest_index( multi_row_dictionary );

if ( dictionary_highest_index() < 1 ) return NULL;

UPDATE_ROW_LIST *update_row_list_calloc(
    void );

LIST *list = list_new();

for (  row_number = 1;
       row_number <= dictionary_highest_index();
       row_number++ )
{
    list_set(
        list,
        update_row_new(
            application_name,
            session_key,
            login_name,
            role_name,
            folder_name,
            multi_row_dictionary,
            file_dictionary,
            relation_one2m_list,
            relation_mto1_isa_list,
            folder_attribute_append_isa_list,
            post_change_process,
            update_root_boolean,
            security_entity,
            appaserver_error_filespecification,
            appaserver_parameter_mount_point,
            row_number ) );
}

if ( list_length( list ) )
{
    int update_row_list_count(
        list /* update_row_list */ );

    void update_row_list_update_changed_list_list_set(
        session_key,
        login_name,
        role_name,
        appaserver_error_filespecification,
        list /* update_row_list in/out */,
        update_row_list_count() );

    int update_row_list_cell_count(
        list /* update_row_list */ );

    LIST *update_row_list_folder_name_list(
        list /* update_row_list */ );

    char *update_row_list_folder_name_list_string(
        char UPDATE_FOLDER_NAME_DELIMITER,
        LIST *update_row_list_folder_name_list() );
}
else
{
    return NULL;
}

/* Usage */
int update_row_list_count(
    LIST *update_row_list );

/* Process */
int list_length( update_row_list );

/* Attributes */
int dictionary_highest_index;
LIST *list;
int count;
int cell_count;
LIST *folder_name_list;
char *folder_name_list_string;
```

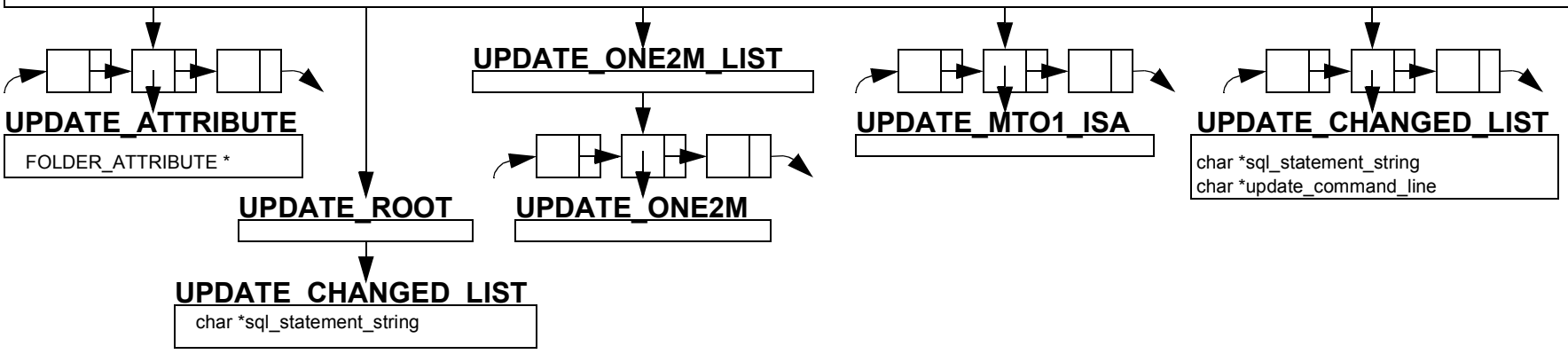


UPDATE_ROW_LIST (2)

```
/* Driver */
/* Returns update_row_list_error_string or NULL */
char *update_row_list_execute(
    const char *SQL_EXECUTABLE,
    char *application_name,
    boolean update_root_boolean,
    UPDATE_ROW_LIST *update_row_list,
    char *update->appaserver_error_filespecification );

/* Process */
char error_string[];
char *ptr = error_string;

for UPDATE_ROW *update_row in update_row_list->list
{
    char *update_error_string =
        char *update_changed_list_list_execute(
            SQL_EXECUTABLE,
            application_name,
            update_root_boolean,
            update_row->update_changed_list_list,
            appaserver_error_filespecification );

    if ( update_error_string )
    {
        if ( ptr != error_string )
        {
            ptr += sprintf( ptr, "\n<br>" );
        }
        ptr += sprintf( ptr, "%s", update_error_string );
    }
}

/* Usage */
void update_row_list_command_line_execute(
    UPDATE_ROW_LIST *update_row_list );

/* Process */
for UPDATE_ROW *update_row in update_row_list->list
{
    for UPDATE_CHANGED_LIST *update_changed_list in
            update_row->update_changed_list_list
    {
        if ( update_changed_list->update_command_line )
        {
            security_system(
                SECURITY_FORK_CHARACTER,
                SECURITY_FORK_STRING,
                update_changed_list->
                    update_command_line );
        }
    }
}

/* Usage */
void update_row_list_display(
    UPDATE_ROW_LIST *update_row_list,
    FILE *output_stream );

/* Process */
for UPDATE_ROW *update_row in update_row_list->list
{
    void update_row_display(
        update_row->update_root,
        update_row->update_one2m_list,
        update_row->update_mto1_isa_list,
        output_stream );
}
```

UPDATE_ROW (1)

```
/* Usage */
UPDATE_ROW *update_row_new(
    char *application_name,
    char *session_key
    char *login_name,
    char *role_name,
    char *folder_name,
    DICTIONARY *multi_row_dictionary,
    DICTIONARY *file_dictionary,
    LIST *relation_one2m_list(
        1 /* include_isa_boolean */ ),
    LIST *relation_mto1_isa_list(),
    LIST *folder_attribute_append_isa_list(),
    PROCESS *post_change_process,
    boolean update_root_boolean,
    SECURITY_ENTITY *security_entity,
    char *appaserver_error_filespecification,
    char *appaserver_parameter_mount_point(),
    int row_number );

/* Process */
UPDATE_ROW *update_row_calloc(
    void );

LIST *update_attribute_list(
    multi_row_dictionary,
    file_dictionary,
    folder_attribute_append_isa_list,
    row_number );

if ( !list_length( update_attribute_list() ) return NULL;

UPDATE_ROOT *update_root =
    update_root_new(
        folder_name,
        folder_attribute_append_isa_list,
        post_change_process,
        security_entity,
        update_attribute_list() );

if (    update_root
&&  update_root->
        update_changed_primary_key_boolean
&&  list_length( relation_one2m_list ) )
{
    UPDATE_ONE2M_LIST *update_one2m_list =
        update_one2m_list_new(
            application_name,
            session_key,
            login_name,
            folder_name,
            role_name,
            row_number,
            relation_one2m_list,
            appaserver_error_filespecification,
            appaserver_parameter_mount_point,
            update_root->update_changed_list );
}

if ( list_length( relation_mto1_isa_list ) )
{
    LIST *update_mto1_isa_list(
        application_name,
        session_key,
        login_name,
        role_name,
        folder_name /* many_folder_name */,
        relation_mto1_isa_list,
        appaserver_error_filespecification,
        appaserver_parameter_mount_point,
        (update_root)
            ? update_root->
                update_changed_primary_key_boolean
            : 0,
        update_attribute_list() );
}

if ( !update_root_boolean ) update_root = null;

if (    !update_root
&&  !update_one2m_list
&&  !list_length( update_mto1_isa_list ) )
{
    return NULL;
}

/* Attributes */
int row_number;
LIST *update_attribute_list;
UPDATE_ROOT *update_root;
UPDATE_ONE2M_LIST *update_one2m_list;
LIST *update_mto1_isa_list;

/* Set externally */
LIST *update_changed_list_list;
```



UPDATE_ROW (2)

```
/* Usage */
int update_row_list_cell_count(
    LIST *update_row_list );

/* Process */
int cell_count = 0;

for UPDATE_ROW *update_row in update_row_list
{
    int update_row_cell_count( update_row );
    cell_count += update_row_cell_count();
}

/* Usage */
int update_row_cell_count(
    UPDATE_ROW *update_row );

/* Process */
int cell_count = 0;

for UPDATE_CHANGED_LIST *update_changed_list in
    update_row->update_changed_list_list
{
    int list_length( update_changed_list->list );
    cell_count += list_length();
}
```

```
/* Usage */
LIST *update_row_list_folder_name_list(
    LIST *update_row_list );

/* Process */
LIST *folder_name_list = list_new();

for UPDATE_ROW *update_row in update_row_list
{
    LIST *update_row_folder_name_list(
        update_row->update_root,
        update_row->update_one2m_list,
        update_row->update_mto1_isa_list );

    list_unique_list(
        folder_name_list /* in/out */,
        update_row_folder_name_list() );
}

/* Usage */
LIST *update_row_folder_name_list(
    UPDATE_ROOT *update_root,
    UPDATE_ONE2M_LIST *update_one2m_list,
    LIST *update_mto1_isa_list );
```

```
/* Process */
LIST *folder_name_list = list_new();

if ( update_root )
{
    list_set(
        folder_name_list,
        update_root->folder_name );
}

if ( update_one2m_list )
{
    list_unique_list(
        folder_name_list /* in/out */,
        update_one2m_list_folder_name_list(
            update_one2m_list->list
                /* update_one2m_list */ ) );
}

if ( list_length( update_mto1_isa_list ) )
{
    list_unique_list(
        folder_name_list /* in/out */,
        update_mto1_isa_list_folder_name_list(
            update_mto1_isa_list );
}
```

UPDATE_ROW (3)

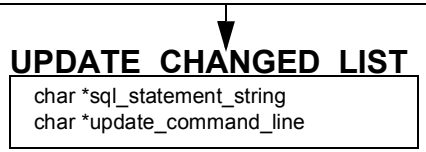
```
/* Usage */
/* Returns update_error_string or null */
char *update_row_execute(
    const char *SQL_EXECUTABLE,
    char *application_name,
    UPDATE_ROOT *update_row->update_root,
    UPDATE_ONE2M_LIST *update_row->
        update_one2m_list,
    LIST *update_row->update_mto1_isa_list,
    char *appaserver_error_filename() );

/* Process */
if ( update_root )
{
    char *update_error_string =
        char *update_root_execute(
            SQL_EXECUTABLE,
            update_root->
                update_changed_list->
                sql_statement_string
                /* update_sql_statement_string */,
            appaserver_error_filename );

    if ( update_error_string )
    {
        appaserver_error_message_file(
            application_name,
            (char *)0 /* login_name */,
            update_error_string /* message */ );

        return update_error_string;
    }
}

char *update_system_string(
    appaserver_error_filename );

FILE *appaserver_output_pipe(
    update_system_string() );

if ( update_one2m_list )
{
    void update_one2m_list_sql_statement_execute(
        update_one2m_list->list
            /* update_one2m_list */,
        appaserver_output_pipe() );
}

if ( list_length( update_mto1_isa_list ) )
{
    void update_mto1_isa_list_sql_statement_execute(
        update_mto1_isa_list,
        appaserver_output_pipe() );
}

pclose( appaserver_output_pipe() );

/* Usage */

/* Execute any post_change_process */
void update_row_command_line_execute(
    UPDATE_ROOT *update_row->update_root,
    UPDATE_ONE2M_LIST *update_row->
        update_one2m_list,
    LIST *update_row->update_mto1_isa_list );

/* Process */
if (    update_root
&&  update_root->update_changed_list
&&  update_root->
        update_changed_list->
        update_command_line )
{
    security_system(
        SECURITY_FORK_CHARACTER,
        SECURITY_FORK_STRING,
        update_root->
            update_changed_list->
            update_command_line );
}

if ( update_one2m_list )
{
    void update_one2m_list_command_line_execute(
        LIST *update_one2m_list->list
            /* update_one2m_list */ );
}

if ( list_length( update_mto1_isa_list ) )
{
    void update_mto1_isa_list_command_line_execute(
        LIST *update_mto1_isa_list );
}
```

UPDATE_ROW (4)

```
/* Usage */
void update_row_display(
    UPDATE_ROOT *update_row->update_root,
    UPDATE_ONE2M_LIST *update_row->
        update_one2m_list,
    LIST *update_row->update_mto1_isa_list,
    FILE *output_stream );

/* Process */
if ( update_root )
{
    fprintf(
        output_stream,
        "%s\n",
        update_root->
            update_changed_list->
            sql_statement_string );
}

if ( update_one2m_list )
{
    void update_one2m_list_display(
        update_one2m_list->list
            /* update_one2m_list */,
        output_stream );
}

if ( list_length( update_mto1_isa_list ) )
{
    void update_mto1_isa_list_display(
        update_mto1_isa_list,
        output_stream );
}

/* Usage */
void update_row_list_update_changed_list_list_set(
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *appaserver_error_filespecification,
    LIST *update_row_list /* in/out */,
    int update_row_list_count );

/* Process */
for UPDATE_ROW *update_row in update_row_list
{
    if (    update_row->update_root
    &&  update_row->update_root->process_command_line )
    {
        char *update_command_line(
            APPASERVER_UPDATE_STATE,
            UPDATE_PREUPDATE_PREFIX,
            update_row->update_root->process_command_line,
            session_key,
            login_name,
            role_name,
            update_row->update_root->folder_name,
            update_row->update_root->process_name,
            appaserver_error_filespecification,
            update_row->
                update_root->
                update_where_list_primary_data_list,
            update_row->update_attribute_list,
            ++row_number,
            update_row_list_count /* row_count */ );

        update_row->
            update_root->
            update_changed_list->
            update_command_line =
                update_command_line();
    }

    LIST *update_row_update_changed_list_list(
        update_row->update_root,
        update_row->update_one2m_list,
        update_row->update_mto1_isa_list );

    update_row->update_changed_list_list =
        update_row_update_changed_list_list();
}

/* Usage */
LIST *update_row_update_changed_list_list(
    UPDATE_ROOT *update_row->update_root,
    UPDATE_ONE2M_LIST *update_row->
        update_one2m_list,
    LIST *update_row->update_mto1_isa_list );

/* Process */
LIST *update_changed_list_list = list_new();

if ( update_root )
{
    list_set(
        update_changed_list_list,
        update_root->
            update_changed_list );
}

if ( update_one2m_list )
{
    void update_one2m_list_update_changed_list_list(
        update_one2m_list->list
            /* update_one2m_list */,
        update_changed_list_list /* in/out */ );
}

if ( list_length( update_mto1_isa_list ) )
{
    void update_mto1_isa_list_update_changed_list_list(
        update_mto1_isa_list,
        update_changed_list_list /* in/out */ );
}
```

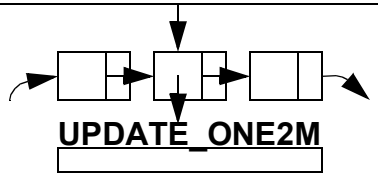
UPDATE_ROOT

```
/* Usage */
UPDATE_ROOT *update_root_new(
    char *folder_name,
    LIST *folder_attribute_append_isa_list,
    PROCESS *post_change_process,
    SECURITY_ENTITY *security_entity,
    LIST *update_attribute_list );

/* Process */
UPDATE_ROOT *update_root_calloc(
    void );

LIST *folder_attribute_primary_key_list(
    folder_name,
    folder_attribute_append_isa_list );

LIST *folder_attribute_name_list(
    folder_name,
    folder_attribute_append_isa_list );

UPDATE_CHANGED_LIST *update_changed_list =
    update_changed_list_new(
        folder_name,
        folder_attribute_primary_key_list(),
        folder_attribute_name_list(),
        update_attribute_list,
        (security_entity)
            ? security_entity->where
            : (char *)0 );

if ( !update_changed_list ) return NULL;

boolean update_changed_primary_key_boolean(
    update_changed_list->list
        /* update_changed_list */ );

if ( post_change_process )
{
    this->process_name =
        post_change_process->process_name;

    this->process_command_line =
        post_change_process->command_line;

    LIST *update_where_list_primary_data_list(
        update_changed_list->where_list
            /* update_where_list */ );
}

/* Usage */

/* Returns char *update_error_string */
char *update_root_execute(
    const char *SQL_EXECUTABLE,
    char *update_changed_list->
        sql_statement_string
        /* update_sql_statement_string */,
    char *appaserver_error_filespecification );

/* Process */
snprintf(
    system_string[],
    sizeof ( system_string ),
    "tee_process.e '%s %s' | %s",
    "update_log.sh",
    appaserver_error_filename,
    SQL_EXECUTABLE );

SPOOL *spool_new(
    system_string,
    1 /* capture_stderr_boolean */ );

fprintf(
    spool_new()->output_pipe,
    "%s\n",
    update_sql_statement_string );

pclose( spool_new()->output_pipe );

LIST *list = spool_list( spool_new()->output_filespecification );
char *update_error_string = list_first( list );

/* Attributes */
char *folder_name;
LIST *folder_attribute_primary_key_list;
LIST *folder_attribute_name_list;
UPDATE_CHANGED_LIST *update_changed_list;
boolean update_changed_primary_key_boolean;
char *process_name;
char *process_command_line;
LIST *update_where_list_primary_data_list
```



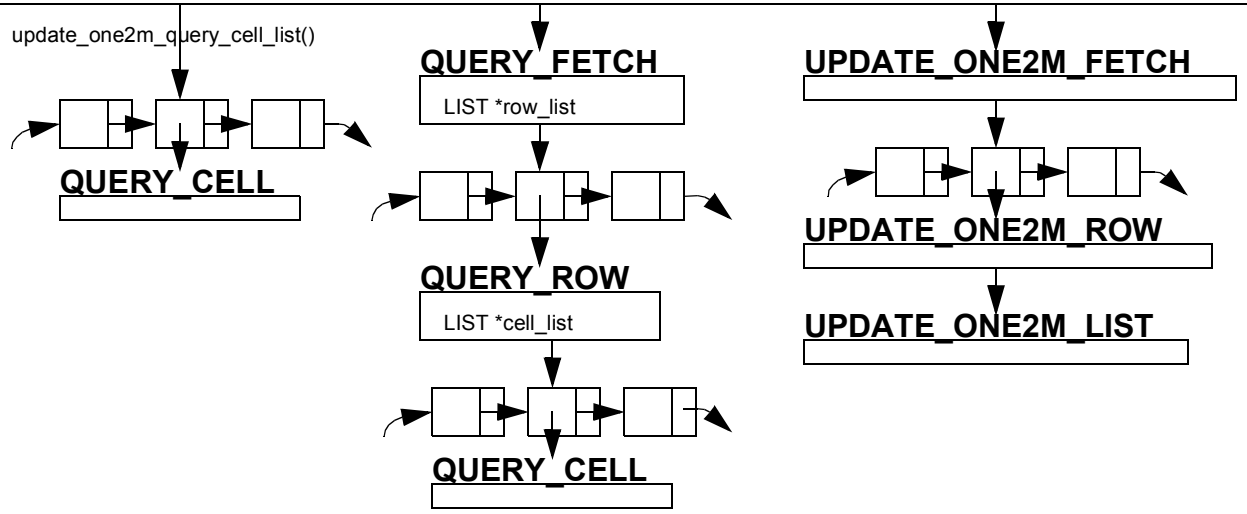
UPDATE_ONE2M_LIST

```
/* Usage */
UPDATE_ONE2M_LIST *update_one2m_list_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *folder_name,
    char *role_name,
    int row_number,
    LIST *relation_one2m_list,
    char *appaserver_error_filespecification,
    char *appaserver_parameter_mount_point,
    UPDATE_CHANGED_LIST *update_changed_list );

/* Process */
UPDATE_ONE2M_LIST *update_one2m_list_calloc(
    void );

LIST *list = list_new();

for RELATION_ONE2M *relation_one2m in
    relation_one2m_list
{
    /* Cache PROCESS *post_change_process */
    if ( relation_one2m->
            many_folder->
            post_change_process_name
    && !relation_one2m->
            many_folder->
            post_change_process )
    {
        relation_one2m->
            many_folder->
            post_change_process =
                PROCESS *process_fetch(
                    relation_one2m->
                        many_folder->
                        post_change_process_name,
                    (char *)0 /* document_root */,
                    (char *)0 /* relative_source */,
                    1 /* check_inside */,
                    appaserver_parameter_mount_point );
    }

    UPDATE_ONE2M *update_one2m =
        update_one2m_new(
            application_name,
            session_key,
            login_name,
            role_name,
            row_number,
            relation_one2m->one_folder_name,
            relation_one2m->many_folder_name,
            relation_one2m->
                many_folder->
                folder_attribute_primary_key_list
                /* many_primary_key_list */,
            relation_one2m->
                many_folder->
                folder_attribute_list
                /* many_folder_attribute_list */,
            relation_one2m->relation_foreign_key_list,
            relation_one2m->relation_translate_list,
            relation_one2m->
                many_folder->
                post_change_process,
            appaserver_error_filespecification,
            appaserver_parameter_mount_point,
            update_changed_list );

        if ( update_one2m )
        {
            list_set(
                list,
                update_one2m );
        }
    }

    if ( !list_length( list ) ) return NULL;

    /* Attributes */
    char *folder_name;
    int row_number;
    LIST *list;
```



UPDATE_ONE2M (1)

```
/* Usage */
UPDATE_ONE2M *update_one2m_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    int row_number,
    char *one_folder_name,
    char *many_folder_name,
    LIST *many_primary_key_list,
    LIST *many_folder_attribute_list,
    LIST *relation_foreign_key_list,
    LIST *relation_translate_list,
    PROCESS *many_post_change_process,
    char *appaserver_error_filename,
    char *appaserver_parameter_mount_point,
    UPDATE_CHANGED_LIST *update_changed_list );

/* Process */
UPDATE_ONE2M *update_one2m_calloc(
    void );

/* Select from both lists in case the foreign key isn't primary */
LIST *update_one2m_select_name_list(
    many_primary_key_list,
    relation_foreign_key_list );

char *update_one2m_select_string(
    update_one2m_select_name_list() );

LIST *update_one2m_where_query_cell_list(
    update_changed_list->where_list
        /* update_where_list */,
    relation_translate_list,
    many_folder_attribute_list );

char *query_cell_where_string(
    update_one2m_where_query_cell_list() );

char *query_system_string(
    application_name,
    update_one2m_select_string(),
    many_folder_name
        /* from_string */,
    query_cell_where_string(),
    (char *)0 /* group_by_string */,
    (char *)0 /* order_string */,
    0 /* max_rows */ );

QUERY_FETCH *query_fetch =
    query_fetch_new(
        many_folder_name,
        many_folder_attribute_list,
        date_convert_international
            /* date_convert_format_enum */,
        update_one2m_select_name_list()
            /* query_select_name_list */,
        query_system_string(),
        1 /* input_save_boolean */ );

if ( !list_length( query_fetch->row_list ) ) return NULL;

UPDATE_ONE2M_FETCH *update_one2m_fetch =
    update_one2m_fetch_new(
        application_name,
        session_key,
        login_name,
        role_name,
        row_number,
        one_folder_name,
        many_folder_name,
        many_primary_key_list,
        many_folder_attribute_list,
        relation_translate_list,
        many_post_change_process,
        appaserver_error_filename,
        appaserver_parameter_mount_point,
        update_changed_list,
        query_fetch->row_list );

/* Attributes */
char *one_folder_name;
char *many_folder_name;
LIST *select_name_list;
char *select_string;
LIST *where_query_cell_list;
char *query_cell_where_string;
char *query_system_string;
QUERY_FETCH *query_fetch;
UPDATE_ONE2M_FETCH *update_one2m_fetch;
```



UPDATE_ONE2M (2)

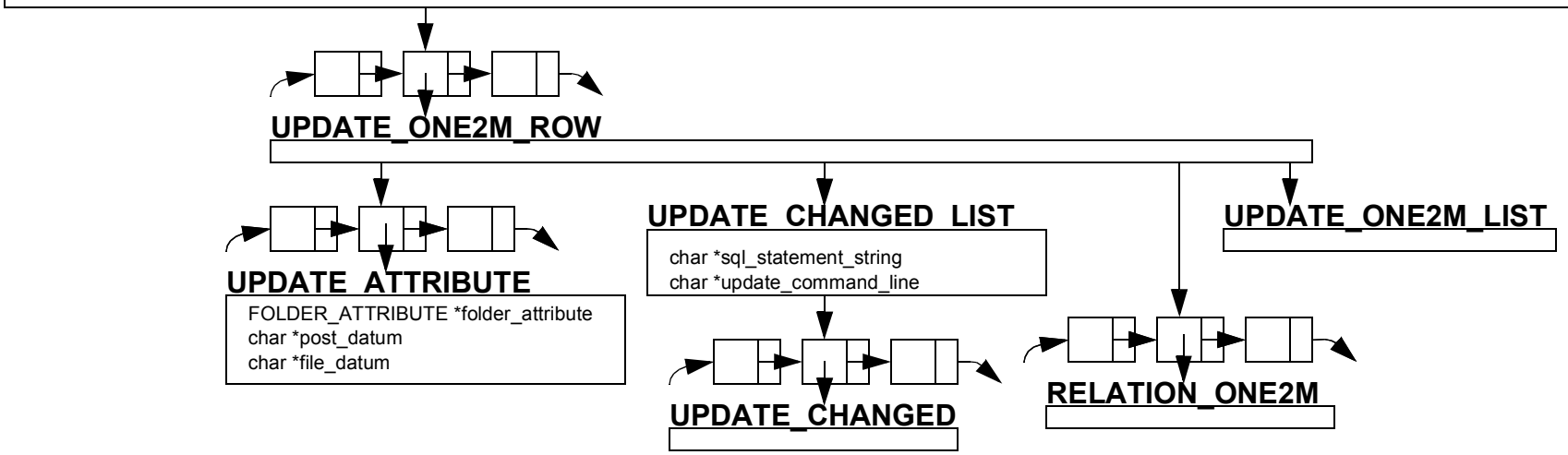
```
/* Usage */
LIST *update_one2m_where_query_cell_list(
    LIST *update_where_list,
    LIST *relation_one2m->relation_translate_list,
    LIST *relation_one2m->many_folder_attribute_list );

/* Process */
LIST *query_cell_list = list_new();

list_rewind( relation_translate_list );

for UPDATE_WHERE *update_where in
    update_where_list
{
    RELATION_TRANSLATE *relation_translate =
        list_get(
            relation_translate_list );

    QUERY_CELL *query_cell_parse(
        many_folder_attribute_list,
        date_convert_international
            /* date_convert_format_enum */,
        relation_translate->
            foreign_key
            /* attribute_name */,
        update_where->
            update_attribute->
            file_datum );

    list_set(
        query_cell_list,
        query_cell_parse() );

    list_next( relation_translate_list );
}

/* Usage */
void update_one2m_list_sql_statement_execute(
    LIST *update_one2m_list,
    FILE *appaserver_output_pipe );

/* Process */
for UPDATE_ONE2M *update_one2m in
    update_one2m_list
{
    void update_one2m_row_list_pipe_execute(
        update_one2m->
            update_one2m_fetch->
            update_one2m_row_list,
        appaserver_output_pipe );
}

/* Usage */
void update_one2m_list_command_line_execute(
    LIST *update_one2m_list );

/* Process */
for UPDATE_ONE2M *update_one2m in
    update_one2m_list
{
    void update_one2m_row_list_command_execute(
        update_one2m->
            update_one2m_fetch->
            update_one2m_row_list );
}

/* Usage */
int update_one2m_list_cell_count(
    LIST *update_one2m_list );

/* Process */
for UPDATE_ONE2M *update_one2m in
    update_one2m_list
{
    cell_count +=
        update_one2m_row_list_cell_count(
            update_one2m->
                update_one2m_fetch->
                update_one2m_row_list );
}

/* Usage */
LIST *update_one2m_list_folder_name_list(
    LIST *update_one2m_list );

/* Process */
LIST *folder_name_list = list_new();

for UPDATE_ONE2M *update_one2m in
    update_one2m_list
{
    list_set(
        folder_name_list,
        update_one2m->many_folder_name );

    if ( update_one2m->update_one2m_fetch )
    {
        LIST *update_one2m_row_list_folder_name_list(
            update_one2m->
                update_one2m_fetch->
                update_one2m_row_list );

        list_unique_list(
            folder_name_list,
            update_one2m_row_list_folder_name_list() );
    }
}

/* Usage */
LIST *update_one2m_select_name_list(
    LIST *many_primary_key_list,
    LIST *relation_foreign_key_list );

/* Process */
LIST *list_copy( many_primary_key_list );

list_set_list( list_copy(), relation_foreign_key_list );

/* Usage */
char *update_one2m_select_string(
    LIST *update_one2m_select_name_list );

/* Process */
char *list_display_delimited(
    update_one2m_select_name_list,
    ',' );
```

UPDATE_ONE2M (3)

```
/* Usage */
void update_one2m_list_display(
    LIST *update_one2m_list,
    FILE *output_stream );

/* Process */
for UPDATE_ONE2M *update_one2m in
    update_one2m_list
{
    void update_one2m_row_list_display(
        update_one2m->
            update_one2m_fetch->
            update_one2m_row_list,
        output_stream );
}

/* Usage */
void update_one2m_list_update_changed_list_list(
    LIST *update_one2m_list,
    LIST *update_changed_list_list /* in/out */ );

/* Process */
for UPDATE_ONE2M *update_one2m in
    update_one2m_list
{
    update_one2m_row_list_update_changed_list_list(
        update_one2m->
            update_one2m_fetch->
            update_one2m_row_list,
        update_changed_list_list /* in/out */ );
}
```

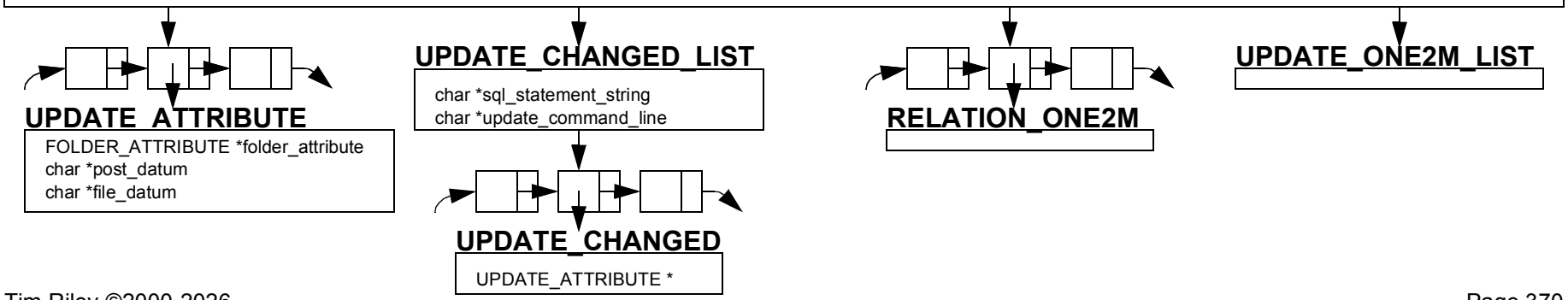
UPDATE_ONE2M_FETCH

```
/* Usage */
UPDATE_ONE2M_FETCH *update_one2m_fetch_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    int row_number,
    char *one_folder_name,
    char *many_folder_name,
    LIST *many_primary_key_list,
    LIST *many_folder_attribute_list,
    LIST *relation_translate_list,
    PROCESS *many_post_change_process,
    char *appaserver_error_filename,
    char *appaserver_parameter_mount_point,
    UPDATE_CHANGED_LIST *update_changed_list,
    LIST *query_fetch_row_list );

/* Process */
UPDATE_ONE2M_FETCH *update_one2m_fetch_calloc(
    void );

LIST *update_one2m_row_list = list_new();

for QUERY_ROW *query_row in
    query_fetch_row_list
{
    UPDATE_ONE2M_ROW *update_one2m_row =
        update_one2m_row_new(
            application_name,
            session_key,
            login_name,
            role_name,
            row_number,
            many_folder_name,
            many_primary_key_list,
            many_folder_attribute_list,
            relation_translate_list,
            many_post_change_process,
            appaserver_error_filename,
            appaserver_parameter_mount_point,
            update_changed_list,
            query_row->cell_list );

    list_set(
        update_one2m_row_list,
        update_one2m_row );
}

/* Attribute */
int row_number;
char *one_folder_name;
char *many_folder_name;
LIST *update_one2m_row_list;
```



UPDATE_ONE2M_ROW (1)

```
/* Usage */
UPDATE_ONE2M_ROW *update_one2m_row_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    int row_number,
    char *many_folder_name,
    LIST *many_primary_key_list,
    LIST *many_folder_attribute_list,
    LIST *relation_translate_list,
    PROCESS *many_post_change_process,
    char *appaserver_error_filespecification,
    char *appaserver_parameter_mount_point,
    UPDATE_CHANGED_LIST *update_changed_list,
    LIST *query_row_cell_list );

/* Process */
UPDATE_ONE2M_ROW *update_one2m_row_calloc(
    void );

LIST *update_one2m_row_update_attribute_list(
    many_folder_name,
    many_primary_key_list,
    many_folder_attribute_list,
    relation_translate_list,
    update_changed_list->list
        /* update_changed_list */,
    query_row_cell_list );

UPDATE_CHANGED_LIST *
    update_one2m_row_update_changed_list(
        many_folder_name,
        many_folder_attribute_list,
        update_one2m_row_update_attribute_list() );

if ( many_post_change_process )
{
    LIST *query_cell_primary_data_list(
        query_row_cell_list );

    char *update_one2m_row_command_line(
        APPASERVER_UPDATE_STATE,
        UPDATE_PREUPDATE_PREFIX,
        many_post_change_process->
            command_line,
        session_key,
        login_name,
        role_name,
        many_folder_name,
        many_post_change_process->
            process_name,
        appaserver_error_filespecification,
        query_cell_primary_data_list(),
        update_one2m_row_update_attribute_list(),
        0 /* row_number */,
        0 /* row_count */ );

    update_one2m_row_update_changed_list()->
        update_command_line =
            update_one2m_row_command_line();
}

boolean update_changed_primary_key_boolean(
    update_changed_list->list
        /* update_changed_list */ );

if ( update_changed_primary_key_boolean() )
{
    LIST *folder_attribute_primary_key_list(
        many_folder_name,
        many_folder_attribute_list );

    LIST *relation_one2m_list(
        many_folder_name,
        folder_attribute_primary_key_list(),
        1 /* include_isa_boolean */ );

    if ( list_length( relation_one2m_list() ) )
    {
        UPDATE_ONE2M_LIST *update_one2m_list =
            update_one2m_list_new(
                application_name,
                session_key,
                login_name,
                many_folder_name,
                role_name,
                relation_one2m_list(),
                appaserver_error_filespecification,
                appaserver_parameter_mount_point,
                update_one2m_row_update_changed_list() );
    }
}

/* Attributes */
int row_number;
char *many_folder_name;
LIST *relation_foreign_key_list;
LIST *update_attribute_list;
UPDATE_CHANGED_LIST *update_changed_list;
LIST *query_cell_primary_data_list;
char *command_line;
boolean update_changed_primary_key_boolean;
LIST *folder_attribute_primary_key_list;
LIST *relation_one2m_list;
UPDATE_ONE2M_LIST *update_one2m_list;
```



UPDATE_ONE2M_ROW (2)

```
/* Usage */
LIST *update_one2m_row_update_attribute_list(
    LIST *many_primary_key_list,
    LIST *many_folder_attribute_list,
    LIST *relation_translate_list,
    LIST *update_changed_list,
    LIST *query_row_cell_list );

/* Process */
LIST *update_attribute_list = list_new();

for QUERY_CELL *query_cell in query_row_cell_list
{
    char *relation_translate_primary_key(
        relation_translate_list,
        query_cell->attribute_name
            /* foreign_key */ );

    char *update_one2m_row_seek_attribute_name(
        char *query_cell->attribute_name,
        char *relation_translate_primary_key() );

    UPDATE_CHANGED *update_changed_seek(
        update_changed_list,
        update_one2m_row_seek_attribute_name() );

    char *update_one2m_row_post_datum(
        query_cell->select_datum,
        update_changed_seek() );

    UPDATE_ATTRIBUTE *update_attribute =
        update_one2m_row_update_attribute(
            many_folder_attribute_list,
            query_cell->attribute_name,
            query_cell->select_datum,
            update_one2m_row_post_datum() );

    list_set(
        update_attribute_list,
        update_attribute );
}

/* Usage */
UPDATE_ATTRIBUTE *
    update_one2m_row_update_attribute(
        LIST *many_folder_attribute_list,
        char *attribute_name,
        char *query_cell_select_datum,
        char *update_one2m_row_post_datum );

/* Process */
UPDATE_ATTRIBUTE *update_attribute_new(
    attribute_name,
    update_one2m_row_post_datum /* post_datum */,
    query_cell_select_datum /* file_datum */ );

FOLDER_ATTRIBUTE *update_attribute->folder_attribute =
    folder_attribute_seek(
        (char *)0 /* folder_name */,
        attribute_name,
        many_folder_attribute_list );

if ( !update_attribute->folder_attribute ) exit( 1 );

/* Usage */
char *update_one2m_row_post_datum(
    char *query_cell_select_datum,
    UPDATE_CHANGED *update_changed_seek );

/* Process */
if ( update_changed_seek )
{
    char *post_datum =
        update_changed_seek->
            update_attribute->
                post_datum;
}
else
{
    post_datum = query_cell_select_datum;
}

/* Usage */
char *update_one2m_row_command_line(
    const char *APPASERVER_UPDATE_STATE,
    const char *UPDATE_PREUPDATE_PREFIX,
    char *post_change_process_command_line,
    char *session_key,
    char *login_name,
    char *role_name,
    char *many_folder_name,
    char *post_change_process_name,
    char *appaserver_error_filespecification,
    LIST *query_cell_primary_data_list,
    LIST *update_one2m_row_update_attribute_list );

/* Process */
char *update_command_line(
    APPASERVER_UPDATE_STATE,
    UPDATE_PREUPDATE_PREFIX,
    post_change_process_command_line,
    session_key,
    login_name,
    role_name,
    many_folder_name,
    post_change_process_name,
    appaserver_error_filespecification,
    query_cell_primary_data_list
        /* update_where_primary_data_list */,
    update_one2m_row_update_attribute_list,
    0 /* row_number */,
    0 /* row_count */ );
```

UPDATE_ONE2M_ROW (3)

```
/* Usage */
int update_one2m_row_list_cell_count(
    LIST *update_one2m_row_list );

/* Process */
int cell_count = 0;

for UPDATE_ONE2M_ROW *update_one2m_row in
    update_one2m_row_list
{
    cell_count +=
        list_length(
            update_one2m_row->
                update_changed_list->
                list /* update_changed_list */ );

    if ( update_one2m_row->update_one2m_list )
    {
        cell_count +=
            update_one2m_list_cell_count(
                update_one2m_row->
                    update_one2m_list->
                    list /* update_one2m_list */ );
    }
}
```

```
/* Usage */
LIST *update_one2m_row_list_folder_name_list(
    LIST *update_one2m_row_list );

/* Process */
LIST *folder_name_list = list_new();

for UPDATE_ONE2M_ROW *update_one2m_row in
    update_one2m_row_list
{
    list_set(
        folder_name_list,
        update_one2m_row->
            update_changed_list->
            folder_name );

    if ( update_one2m_row->update_one2m_list )
    {
        LIST *update_one2m_list_folder_name_list(
            update_one2m_row->
                update_one2m_list->
                list /* update_one2m_list */ );

        list_unique_list(
            folder_name_list,
            update_one2m_list_folder_name_list() );
    }
}
```

```
/* Usage */
UPDATE_CHANGED_LIST *
    update_one2m_row_update_changed_list(
        char *many_folder_name,
        LIST *many_folder_attribute_list,
        LIST *update_one2m_row->update_attribute_list );

/* Process */
LIST *folder_attribute_primary_key_list(
    many_folder_name,
    many_folder_attribute_list );

LIST *folder_attribute_name_list(
    many_folder_name,
    many_folder_attribute_list );

UPDATE_CHANGED_LIST *update_changed_list_new(
    many_folder_name,
    folder_attribute_primary_key_list(),
    folder_attribute_name_list(),
    update_attribute_list,
    (char *)0 /* security_entity_where */ );
```

UPDATE_ONE2M_ROW (4)

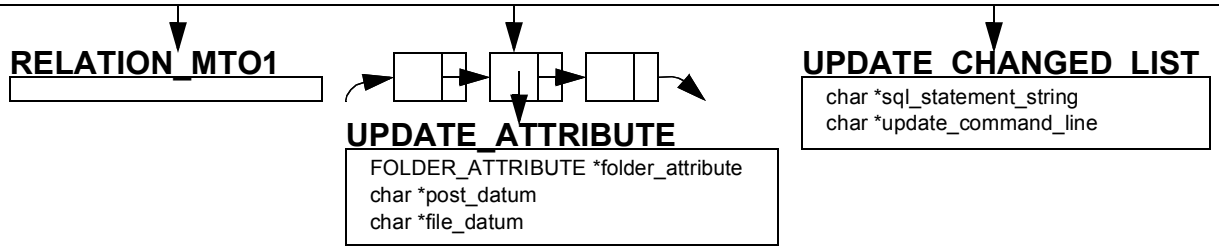
```
/* Usage */
void update_one2m_row_list_pipe_execute(
    LIST *update_one2m_row_list,
    FILE *appaserver_output_pipe );

/* Process */
for UPDATE_ONE2M_ROW *update_one2m_row in
    update_one2m_row_list
{
    fprintf(
        appaserver_output_pipe,
        "%s\n",
        update_one2m_row->
            update_changed_list->
            sql_statement_string );

    if ( update_one2m_row->update_one2m_list )
    {
        update_one2m_list_sql_statement_execute(
            update_one2m_row->
                update_one2m_list->
                list /* update_one2m_list */,
            appaserver_output_pipe );
    }
}

/* Usage */
void update_one2m_row_list_command_execute(
    LIST *update_one2m_row_list );

/* Process */
for UPDATE_ONE2M_ROW *update_one2m_row in
    update_one2m_row_list
{
    if ( update_one2m_row->command_line )
    {
        security_system(
            SECURITY_FORK_CHARACTER,
            SECURITY_FORK_STRING,
            update_one2m_row->command_line );
    }

    if ( update_one2m_row->update_one2m_list )
    {
        update_one2m_list_command_line_execute(
            update_one2m_row->
                update_one2m_list->
                list
                /* update_one2m_list */ );
    }
}

/* Usage */
void update_one2m_row_list_display(
    LIST *update_one2m_row_list,
    FILE *output_stream );

/* Process */
for UPDATE_ONE2M_ROW *update_one2m_row in
    update_one2m_row_list
{
    fprintf(
        output_stream,
        "%s\n",
        update_one2m_row->
            update_changed_list->
            sql_statement_string );

    if ( update_one2m_row->update_one2m_list )
    {
        update_one2m_list_display(
            update_one2m_row->
                update_one2m_list->
                list
                /* update_one2m_list */,
            output_stream );
    }
}

/* Usage */
void update_one2m_row_list_update_changed_list_list(
    LIST *update_one2m_row_list,
    LIST *update_changed_list_list /* in/out */ );

/* Process */
for UPDATE_ONE2M_ROW *update_one2m_row in
    update_one2m_row_list
{
    update_changed_list_list_getset(
        update_changed_list_list /* in/out */,
        update_one2m_row->
            update_changed_list );

    if ( update_one2m_row->update_one2m_list )
    {
        update_one2m_list_update_changed_list_list(
            update_one2m_row->
                update_one2m_list->
                list /* update_one2m_list */,
            update_changed_list_list /* in/out */ );
    }
}
```

UPDATE_MTO1_ISA (1)

```
/* Usage */
LIST *update_mto1_isa_list(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *many_folder_name,
    int row_number,
    LIST *relation_mto1_isa_list,
    char *appaserver_error_filespecification(),
    char *appaserver_parameter_mount_point(),
    boolean update_changed_primary_key_boolean(),
    LIST *update_attribute_list() );

/* Process */
LIST *list = list_new();

for RELATION_MTO1 *relation_mto1_isa in
    relation_mto1_isa_list
{
    UPDATE_MTO1_ISA *update_mto1_isa =
        update_mto1_isa_new(
            application_name,
            session_key,
            login_name,
            role_name,
            many_folder_name,
            row_number,
            relation_mto1_isa,
            appaserver_error_filespecification,
            appaserver_parameter_mount_point,
            update_changed_primary_key_boolean,
            update_attribute_list );

    if ( update_mto1_isa )
    {
        list_set( list, update_mto1_isa );
    }
}

if ( !list_length( list ) ) return NULL;
```

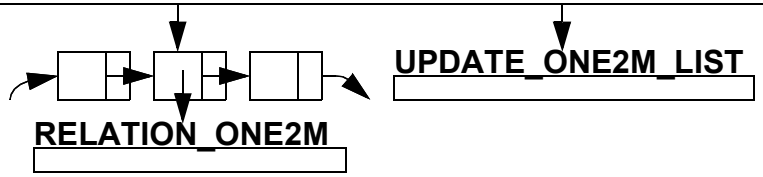
UPDATE_MTO1_ISA (2)

```
/* Usage */
UPDATE_MTO1_ISA *update_mto1_isa_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *many_folder_name,
    int row_number,
    RELATION_MTO1 *relation_mto1_isa,
    char *appaserver_error_filespecification,
    char *appaserver_parameter_mount_point,
    boolean update_changed_primary_key_boolean,
    LIST *update_attribute_list );

/* Process */
UPDATE_MTO1_ISA *update_mto1_isa_calloc(
    void );

LIST *update_mto1_isa_update_attribute_list(
    many_folder_name,
    relation_mto1_isa->
        one_folder->
        folder_attribute_list,
    relation_mto1_isa->relation_translate_list,
    update_attribute_list );

UPDATE_CHANGED_LIST *update_changed_list =
    update_changed_list_new(
        application_name,
        relation_mto1_isa->one_folder_name,
        relation_mto1_isa->
            one_folder->
            folder_attribute_primary_key_list,
        relation_mto1_isa->
            one_folder->
            folder_attribute_name_list,
        update_mto1_isa_update_attribute_list(),
        (char *)0 /* security_entity_where */ );

if ( !update_changed_list ) return NULL;

if (    relation_mto1_isa->
            one_folder->
            post_change_process_name
&&  !relation_mto1_isa->
            one_folder->
            post_change_process )
{
    relation_mto1_isa->
        one_folder->
        post_change_process =
            PROCESS *process_fetch(
                relation_mto1_isa->
                    one_folder->
                    post_change_process_name,
                (char *)0 /* document_root */,
                (char *)0 /* relative_source */,
                1 /* check_executable */,
                appaserver_parameter_mount_point );
}

if ( relation_mto1_isa->
        one_folder->
        post_change_process )
{
    LIST *update_where_list_primary_data_list(
        update_changed_list->where_list
            /* update_where_list */ );

    char *update_command_line(
        APPASERVER_UPDATE_STATE,
        UPDATE_PREUPDATE_PREFIX,
        relation_mto1_isa->
            one_folder->
            post_change_process->
            command_line,
        session_key,
        login_name,
        role_name,
        relation_mto1_isa->one_folder_name,
        relation_mto1_isa->
            one_folder->
            post_change_process_name,
        appaserver_error_filename,
        update_where_list_primary_data_list(),
        update_mto1_list_update_attribute_list(),
        0 /* row_number */,
        0 /* row_count */ );

    update_changed_list->update_command_line =
        update_command_line();
}
```



UPDATE_MTO1_ISA (3)

```
/* Process (continued) */
if ( update_changed_primary_key_boolean )
{
    LIST *relation_one2m_list(
        relation_mto1_isa->one_folder_name,
        relation_mto1_isa->
            one_folder->
            folder_attribute_primary_key_list,
        1 /* include_isa_boolean */ );

    if ( list_length( relation_one2m_list() )
    {
        UPDATE_ONE2M_LIST *update_one2m_list =
            update_one2m_list_new(
                application_name,
                session_key,
                login_name,
                relation_mto1_isa->one_folder_name,
                role_name,
                row_number,
                relation_one2m_list(),
                appaserver_error_filespecification,
                appaserver_parameter_mount_point,
                update_changed_list );
    }
}

/* Attributes */
RELATION_MTO1 *relation_mto1_isa;
LIST *update_attribute_list;
UPDATE_CHANGED_LIST *update_changed_list;
LIST *update_where_list_primary_data_list;
LIST *relation_one2m_list;
UPDATE_ONE2M_LIST *update_one2m_list;
```



UPDATE_MTO1_ISA (4)

```
/* Usage */
LIST *update_mto1_isa_update_attribute_list(
    char *many_folder_name,
    LIST *one_folder_attribute_list,
    LIST *relation_mto1_isa->relation_translate_list,
    LIST *update_attribute_list );

/* Process */
LIST *list = list_new();

for FOLDER_ATTRIBUTE *folder_attribute in
    one_folder_attribute_list
{
    UPDATE_ATTRIBUTE *
        update_mto1_isa_update_attribute(
            many_folder_name,
            folder_attribute
                /* one_folder_attribute */
            relation_translate_list,
            update_attribute_list );

    if ( update_mto1_isa_update_attribute() )
    {
        list_set(
            list,
            update_mto1_isa_update_attribute() );
    }
}

/* Usage */
UPDATE_ATTRIBUTE *
    update_mto1_isa_update_attribute(
        char *many_folder_name,
        FOLDER_ATTRIBUTE *one_folder_attribute,
        LIST *relation_translate_list,
        LIST *update_attribute_list );

/* Process */
if ( one_folder_attribute->primary_key_index )
{
    char *folder_name = many_folder_name;

    char *attribute_name =
        relation_translate_foreign_key(
            relation_translate_list,
            one_folder_attribute->attribute_name
                /* primary_key */ );
}
else
{
    char *folder_name =
        one_folder_attribute->folder_name;

    char *attribute_name =
        one_folder_attribute->attribute_name;
}

UPDATE_ATTRIBUTE *update_attribute_seek(
    folder_name,
    attribute_name,
    update_attribute_list );

if ( !update_attribute_seek() ) return NULL;

UPDATE_ATTRIBUTE *update_attribute_new(
    one_folder_attribute->attribute_name,
    update_attribute_seek()->post_datum,
    update_attribute_seek()->file_datum );

update_attribute_new()->folder_attribute =
    one_folder_attribute;

/* Usage */
int update_mto1_isa_list_cell_count(
    LIST *update_mto1_isa_list() );

/* Process */
for UPDATE_MTO1_ISA *update_mto1_isa in
    update_mto1_isa_list
{
    cell_count +=
        update_mto1_isa_cell_count(
            update_mto1_isa );
}

/* Usage */
int update_mto1_isa_cell_count(
    UPDATE_MTO1_ISA *update_mto1_isa );

/* Process */
int cell_count =
    list_length(
        update_mto1_isa->
            update_changed_list->
                changed_list );

if ( update_mto1_isa->update_one2m_list )
{
    cell_count +=
        update_one2m_list_cell_count(
            update_mto1_isa->update_one2m_list->list
                /* update_one2m_list */ );
}

/* Usage */
LIST *update_mto1_isa_list_folder_name_list(
    LIST *update_mto1_isa_list() );

/* Process */
LIST *folder_name_list = list_new();

for UPDATE_MTO1_ISA *update_mto1_isa in
    update_mto1_isa_list
{
    list_unique_list(
        folder_name_list,
        update_mto1_isa_folder_name_list(
            update_mto1 ) );
}

/* Usage */
LIST *update_mto1_isa_folder_name_list(
    UPDATE_MTO1_ISA *update_mto1_isa );

/* Process */
LIST *folder_name_list = list_new();

list_set(
    folder_name_list,
    update_mto1_isa->
        relation_mto1_isa->
        one_folder_name );

if ( update_mto1_isa->update_one2m_list )
{
    list_unique_list(
        folder_name_list /* in/out */,
        update_one2m_list_folder_name_list(
            update_mto1_isa->update_one2m_list->list
                /* update_one2m_list */ ) );
}
```

UPDATE_MTO1_ISA (5)

```
/* Usage */
void update_mto1_isa_list_sql_statement_execute(
    LIST *update_mto1_isa_list,
    FILE *appaserver_output_pipe() );

/* Process */
for UPDATE_MTO1_ISA *update_mto1_isa in
    update_mto1_isa_list
{
    void update_mto1_isa_sql_statement_execute(
        update_mto1_isa,
        appaserver_output_pipe );
}

/* Usage */
void update_mto1_isa_sql_statement_execute(
    UPDATE_MTO1_ISA *update_mto1_isa,
    FILE *appaserver_output_pipe() )

/* Process */
fprintf(
    appaserver_output_pipe,
    "%s\n",
    update_mto1_isa->
        update_changed_list->
        sql_statement_striing );

if ( update_mto1_isa->update_one2m_list )
{
    for UPDATE_ONE2M *update_one2m in
        update_mto1_isa->update_one2m_list->list
    {
        void update_one2m_row_list_pipe_execute(
            update_one2m->
                update_one2m_fetch->
                update_one2m_row_list,
            appaserver_output_pipe );
    }
}

/* Usage */
void update_mto1_isa_list_command_line_execute(
    LIST *update_mto1_isa_list );

/* Process */
for UPDATE_MTO1_ISA *update_mto1_isa in
    update_mto1_isa_list
{
    void update_mto1_isa_command_line_execute(
        update_mto1_isa );
}

/* Usage */
void update_mto1_isa_command_line_execute(
    UPDATE_MTO1_ISA *update_mto1_isa );

/* Process */
if ( update_mto1_isa->
        update_changed_list->
        command_line )
    security_system(
        SECURITY_FORK_CHARACTER,
        SECURITY_FORK_STRING,
        update_mto1_isa->
            update_changed_list->
            command_line );

if ( update_mto1_isa->update_one2m_list )
{
    for UPDATE_ONE2M *update_one2m in
        update_mto1_isa->update_one2m_list->list
    {
        void update_one2m_row_list_command_execute(
            update_one2m->
                update_one2m_fetch->
                update_one2m_row_list );
    }
}
```

UPDATE_MTO1_ISA (6)

```
/* Usage */
void update_mto1_isa_list_display(
    LIST *update_mto1_isa_list,
    FILE *output_stream );

/* Process */
for UPDATE_MTO1_ISA *update_mto1_isa in
    update_mto1_isa_list
{
    void update_mto1_isa_display(
        update_mto1_isa,
        output_stream );
}

/* Usage */
void update_mto1_isa_display(
    UPDATE_MTO1_ISA *update_mto1_isa,
    FILE *output_stream );

/* Process */
fprintf(
    output_stream,
    "%s\n",
    update_mto1_isa->
        update_changed_list->
        sql_statement_striing );

if ( update_mto1_isa->update_one2m_list )
{
    for UPDATE_ONE2M *update_one2m in
        update_mto1_isa->update_one2m_list->list
    {
        void update_one2m_row_list_display(
            update_one2m->
                update_one2m_fetch->
                update_one2m_row_list,
            output_stream );
    }
}

/* Usage */
void update_mto1_isa_list_update_changed_list_list(
    LIST *update_mto1_isa_list,
    LIST *update_changed_list_list /* in/out */ );

/* Process */
for UPDATE_MTO1_ISA *update_mto1_isa in
    update_mto1_isa_list
{
    void update_mto1_isa_update_changed_list_list(
        update_mto1_isa,
        update_changed_list_list /* in/out */ );
}

/* Usage */
void update_mto1_isa_update_changed_list_list(
    UPDATE_MTO1_ISA *update_mto1_isa,
    LIST *update_changed_list_list /* in/out */ );

/* Process */
void update_changed_list_list_getset(
    update_changed_list_list /* in/out */,
    update_mto1_isa->
        update_changed_list );

if ( update_mto1_isa->update_one2m_list )
{
    for UPDATE_ONE2M *update_one2m in
        update_mto1_isa->update_one2m_list->list
    {
        update_one2m_row_list_update_changed_list_list(
            update_one2m->
                update_one2m_fetch->
                update_one2m_row_list,
            update_changed_list_list /* in/out */ );
    }
}
```

UPDATE_ATTRIBUTE (1)

```
/* Usage */
LIST *update_attribute_list(
    DICTIONARY *multi_row_dictionary,
    DICTIONARY *file_dictionary,
    LIST *folder_attribute_append_isa_list,
    int row_number );

/* Process */
LIST *list = list_new();

for FOLDER_ATTRIBUTE *folder_attribute in
    folder_attribute_append_isa_list
{
    UPDATE_ATTRIBUTE *update_attribute_resolve_new(
        multi_row_dictionary,
        file_dictionary,
        row_number,
        folder_attribute );

    if ( update_attribute_resolve_new() )
    {
        list_set( list, update_attibute_resolve_new() );
    }
}
```

```
/* Usage */
UPDATE_ATTRIBUTE *update_attribute_resolve_new(
    DICTIONARY *multi_row_dictionary,
    DICTIONARY *file_dictionary,
    int row_number,
    FOLDER_ATTRIBUTE *folder_attribute );

/* Process */
char *widget_container_key(
    folder_attribute->attribute_name
        /* widget_name */,
    row_number );

char *post_datum =
    dictionary_get(
        widget_container_key(),
        multi_row_dictionary );

if ( !post_datum ) return NULL;

char *file_datum =
    dictionary_get(
        widget_container_key(),
        file_dictionary );
```

```
UPDATE_ATTRIBUTE *update_attribute_new(
    folder_attribute->attribute_name,
    post_datum,
    file_datum );

/* Usage */
UPDATE_ATTRIBUTE *update_attribute_new(
    char *attribute_name,
    char *post_datum,
    char *file_datum );

/* Process */
UPDATE_ATTRIBUTE *update_attribute_calloc(
    void );

/* Attributes */
char *attribute_name;
char *post_datum;
char *file_datum;
FOLDER_ATTRIBUTE *folder_attribute;
```

UPDATE_ATTRIBUTE (2)

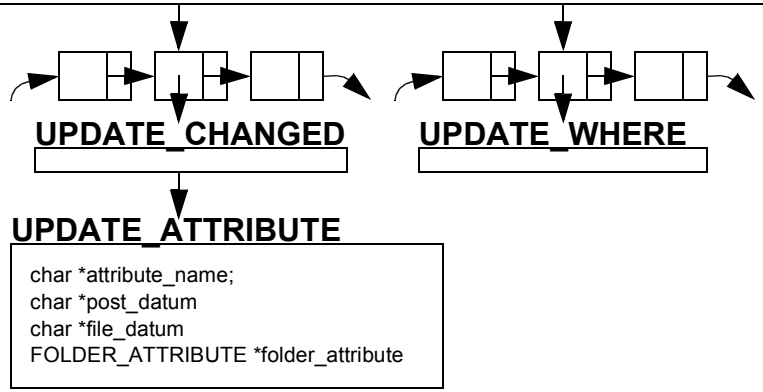
```
/* Usage */
UPDATE_ATTRIBUTE *update_attribute_seek(
    char *folder_name,
    char *attribute_name,
    LIST *update_attribute_list );

/* Process */
for UPDATE_ATTRIBUTE *update_attribute in
    update_attribute_list
{
    if (    folder_name
    &&  update_attribute->folder_attribute )
    {
        if (string_strcmp(
                update_attribute->
                    folder_attribute->
                    folder_name,
                folder_name ) != 0 )
        {
            continue;
        }
    }

    if (string_strcmp(
            update_attribute->attribute_name,
            attribute_name ) == 0 )
    {
        return this;
    }
}
return NULL;
```

```
/* Usage */
char *update_attribute_list_display(
    LIST *update_attibute_list() );

/* Process */
char display[],
char *ptr = display;

for UPDATE_ATTRIBUTE *update_attribute in
    update_attribute_list
{
    char *update_attribute_display(
        char *update_attribute->
            folder_attribute->
            folder_name,
        char *update_attribute->
            folder_attribute->
            attribute_name,
        char *update_attribute->
            post_datum,
        char *update_attribute->
            file_datum );

    ptr += sprintf( ptr,
        "%s\n",
        update_attribute_display() );
}

/* Usage */
char *update_attribute_append_time_post_datum(
        char *post_datum,
        int folder_attribute_width );
```

```
/* Process */
boolean string_character_exists(
    post_datum, ' ' );

if ( string_character_exists() return post_datum;

if ( folder_attribute_width == 16 )
{
    char *time_string =
        date_now_time_hhmm(
            date_utc_offset() );
}
else
if ( folder_attribute_width == 19 )
{
    char *time_string =
        date_now_time_second(
            date_utc_offset() );
}
else
{
    exit( 1 );
}

snprintf(
    append_time_post_datum[],
    sizeof ( append_time_post_datum ),
    "%s %s",
    post_datum,
    time_string );
```

UPDATE_CHANGED_LIST (1)

```
/* Usage */
UPDATE_CHANGED_LIST *update_changed_list_new(
    char *folder_name,
    LIST *folder_attribute_primary_key_list,
    LIST *folder_attribute_name_list,
    LIST *update_attribute_list,
    char *security_entity_where );

/* Process */
UPDATE_CHANGED_LIST *
    update_changed_list_calloc(
        void );

for char *attribute_name in
    folder_attribute_name_list
{
    UPDATE_CHANGED *update_changed =
        update_changed_new(
            folder_name,
            attribute_name,
            update_attribute_list );

    if ( update_changed )
    {
        if ( !changed_list ) list = list_new();
        list_set(
            list,
            update_changed );
    }
}

if ( !list_length( list ) ) return NULL;

char *update_changed_list_set_string(
    list /* changed_list */ );

LIST *where_list = list_new();

for char *primary_key in
    folder_attribute_primary_key_list
{
    UPDATE_WHERE *update_where_new(
        folder_name,
        primary_key,
        update_attribute_list );

    if ( !update_where_new() ) exit( 1 );

    list_set(
        where_list,
        update_where_new() );
}

char *update_where_list_string(
    security_entity_where,
    where_list );

char *appaserver_table_name(
    folder_name );

char *update_changed_list_sql_statement_string(
    appaserver_table_name(),
    update_changed_list_set_string(),
    update_where_list_string() );

/* Attributes */
char *folder_name;
LIST *list;
char *set_string;
LIST *where_list;
char *update_where_list_string;
char *sql_statement_string;

/* Set externally */
char *update_command_line;
```



UPDATE_CHANGED_LIST (2)

```
4/* Usage */
char *update_changed_list_set_string(
    LIST *changed_list );

/* Process */
char set_string[];
char *ptr = set_string;

for UPDATE_CHANGED *update_changed in
    changed_list
{
    if ( ptr != set_string ) ptr += sprintf( ptr, "," );

    ptr += sprintf( ptr,
           "%s",
           update_changed->set_string );
}

/* Usage */
char *update_changed_list_sql_statement_string(
    char *table_name,
    char *set_string,
    char *where_list_string );

/* Process */
snprintf(
    sql_statement_string[],
    sizeof ( sql_statement_string ),
    "update %s set %s where %s;",
    table_name,
    set_string,
    where_list_string );

/* Usage */
char *update_changed_list_display(
    UPDATE_CHANGED_LIST *update_changed_list );

/* Process */
char display[];
char *ptr = display;

for UPDATE_CHANGED *update_changed in
    update_changed_list->list
{
    char *update_changed_display(
        update_changed );

    ptr += sprintf( ptr,
        "[%s]\n",
        update_changed_display() );
}

/* Usage */
void update_changed_list_list_getset(
    LIST *update_changed_list_list /* in/out */,
    UPDATE_CHANGED_LIST *
        parameter_update_changed_list );

/* Process */
for UPDATE_CHANGED_LIST *update_changed_list in
    update_changed_list_list
{
    boolean update_changed_list_boolean(
        update_changed_list,
        parameter_update_changed_list );

    if ( update_changed_list_boolean() ) return;
}

list_set(
    update_changed_list_list,
    parameter_update_changed_list );

/* Usage */
boolean update_changed_list_boolean(
    UPDATE_CHANGED_LIST *update_changed_list,
    UPDATE_CHANGED_LIST *
        parameter_update_changed_list );

/* Process */
boolean list_boolean = 0;

int strcmp(
        update_changed_list->
            sql_statement_string,
        parameter_update_changed_list->
            sql_statement_string );

if ( strcmp() == 0 ) list_boolean = 1;

/* Usage */
/* Returns update_error_string or null */
char *update_changed_list_list_execute(
    const char *SQL_EXECUTABLE,
    char *application_name,
    boolean update_root_boolean,
    LIST *update_changed_list_list,
    char *appaserver_error_filespecification,
    boolean update_root_boolean );

/* Process */
char *update_system_string(
    SQL_EXECUTABLE,
    appaserver_error_filespecification );

FILE *appaserver_output_pipe(
    update_system_string() );

for UPDATE_CHANGED_LIST *update_changed_list in
    update_changed_list_list
{
    if (    update_root_boolean
    &&  list_first_boolean( update_changed_list_list ) )
    {
        char *update_error_string =
            update_root_execute(
                SQL_EXECUTABLE,
                update_changed_list->
                    sql_statement_string,
                appaserver_error_filespecification );

        if ( update_error_string )
        {
            appaserver_error_message_file(
                application_name,
                (char *)0 /* login_name */,
                update_error_string /* message */ );

            pclose( appaserver_output_pipe() );
            return update_error_string;
        }
    }
    else
    {
        fprintf(
            appaserver_output_pipe(),
            "%s\n",
            update_changed_list->sql_statement_string );
    }
}
pclose( appaserver_output_pipe() );
```

UPDATE_CHANGED (1)

```
/* Usage */
UPDATE_CHANGED *update_changed_new(
    char *folder_name,
    char *attribute_name,
    LIST *update_attribute_list );

/* Process */
UPDATE_CHANGED *update_changed_calloc(
    void );

UPDATE_ATTRIBUTE *update_attribute =
    update_attribute_seek(
        folder_name,
        attribute_name,
        update_attribute_list );

if ( !update_attribute ) return NULL;

boolean update_changed_boolean(
    QUERY_IS_NULL,
    update_attribute->file_datum,
    update_attribute->post_datum );

if ( !update_changed_boolean() ) return NULL;

char *update_changed_set_string(
    QUERY_IS_NULL,
    update_attribute->
        folder_attribute->
        attribute_name,
    update_attribute->
        folder_attribute->
        attribute->
        datatype_name,
    update_attribute->
        post_data );

/* Attributes */
UPDATE_ATTRIBUTE *update_attribute;
char *set_string;
```

UPDATE_ATTRIBUTE

```
char *attribute_name;
char *post_datum
char *file_datum
FOLDER_ATTRIBUTE *folder_attribute
```

UPDATE_CHANGED (2)

```
/* Usage */
boolean update_changed_boolean(
    const char *QUERY_IS_NULL,
    char *update_attribute->file_datum,
    char *update_attribute->post_datum );

/* Process */
if (    ( !file_datum || !*file_datum )
&&  !*post_datum )
{
    return 0;
}

if (    !*file_datum
&&  strcmp( post_datum, QUERY_IS_NULL ) == 0 )
{
    return 0;
}

char *security_sql_injection_unescape(
    SECRUITY_ESCAPE_CHARACTER_STRING,
    post_datum );

if ( strcmp(
        security_sql_injection_unescape(),
        file_datum ) != 0 ) )
{
    changed_boolean = 1;
}

/* Usage */
char *update_changed_set_string(
    const char *QUERY_IS_NULL,
    char *attribute_name,
    char *datatype_name,
    char *post_datum );

/* Process */
if (    !*post_datum
||      strcmp( post_datum, "/" ) == 0
||      strcmp( post_datum, QUERY_IS_NULL ) == 0 )
{
    snprintf(
        set_string[],
        sizeof ( set_string ),
        "%s = null",
        attribute_name );
}
else
if ( attribute_is_number( datatype_name ) )
    char *format_string = "%s = %s";
else
    char *format_string = "%s = '%s'";

if ( !*set_string )
{
    snprintf(
        set_string[],
        sizeof ( set_string ),
        format_string,
        attribute_name,
        post_datum );
}

/* Usage */
boolean update_changed_primary_key_boolean(
    LIST *update_changed_list );

/* Process */
for UPDATE_CHANGED *update_changed in
    update_changed_list
{
    if ( update_changed->
        folder_attribute->
        primary_key_index )
    {
        return 1;
    }
}
return 0;

/* Usage */
UPDATE_CHANGED *update_changed_seek(
    LIST *update_changed_list,
    char *attribute_name );

/* Process */
for UPDATE_CHANGED *update_changed in
    update_changed_list
{
    int strcmp(
            update_changed->
                folder_attribute->
                attribute_name,
            attribute_name );

    if ( strcmp() == 0 ) return this;
}
return NULL;

/* Usage */
char *update_changed_display(
    UPDATE_CHANGED *update_changed );

/* Process */
char display[];
char *ptr = display;

ptr += sprintf( ptr,
    "folder=%s, "
    "attribute=%s, "
    "primary=%d, "
    "post_datum=%s, "
    "file_datum=%s",
    update_changed->
        update_attribute->
        folder_attribute->
        folder_name,
    update_changed->
        update_attribute->
        folder_attribute->
        attribute_name,
    update_changed->
        update_attribute->
        folder_attribute->
        primary_key_index,
    update_changed->
        update_attribute->
        post_datum,
    update_changed->
        update_attribute->
        file_datum );
```

UPDATE_WHERE

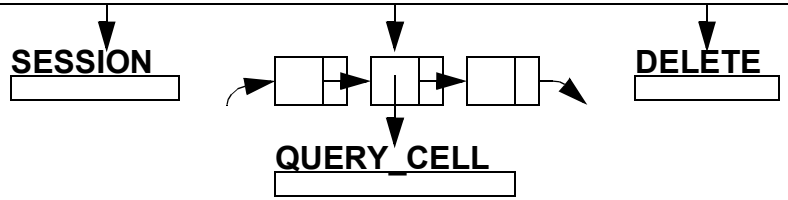
```
/* Usage */
UPDATE_WHERE *update_where_new(
    char *folder_name,
    char *primary_key,
    LIST *update_attribute_list );

/* Process */
UPDATE_WHERE *update_where_calloc(
    void );

UPDATE_ATTRIBUTE *update_attribute =
    update_attribute_seek(
        folder_name,
        primary_key /* attribute_name */,
        update_attribute_list );

if ( !update_attribute ) return null;

char *update_where_string(
    primary_key,
    update_attribute->
        folder_attribute->
        attribute->
        datatype_name,
    update_attribute->
        file_datum );

/* Usage */
char *update_where_list_string(
    char *security_entity_where,
    LIST *update_where_list );

/* Process */
char string[];
char *ptr = string;

for UPDATE_WHERE *update_where in
    update_where_list
{
    if ( ptr != string ) ptr += sprintf( ptr, " and ");
    ptr +=
        sprintf(
            ptr,
            "%s",
            update_where->string );
}

if ( security_entity_where )
{
        sprintf( string + strlen( string ),
            " and %s",
            security_entity_where );
}

/* Usage */
char *update_where_string(
    char *primary_key,
    char *datatype_name,
    char *datum );

/* Process */
boolean attribute_is_number(
    datatype_name );

if ( attribute_is_number() )
    char *format_string = "%s = %s";
else
    char *format_string = "%s = '%s'";

char *security_sql_injection_escape(
    SECURITY_ESCAPE_CHARACTER_STRING,
    datum );

snprintf(
    string[],
    sizeof ( string ),
    format_string,
    primary_key,
    security_sql_injection_escape() );

/* Usage */
LIST *update_where_list_primary_data_list(
    LIST *update_where_list );

/* Process */
for UPDATE_WHERE *update_where in
    update_where_list
{
    boolean update_changed_boolean(
        QUERY_IS_NULL,
        update_where->
            update_attribute->
            file_datum,
        update_where->
            update_attribute->
            post_datum );

    if ( update_changed_boolean() )
    {
        list_set(
            primary_data_list,
            update_where->
                update_attribute->
                post_datum );
    }
    else
    {
        list_set(
            primary_data_list,
            update_where->
                update_attribute->
                file_datum );
    }
}

/* Attributes */
UPDATE_ATTRIBUTE *update_attribute;
char *string;
```

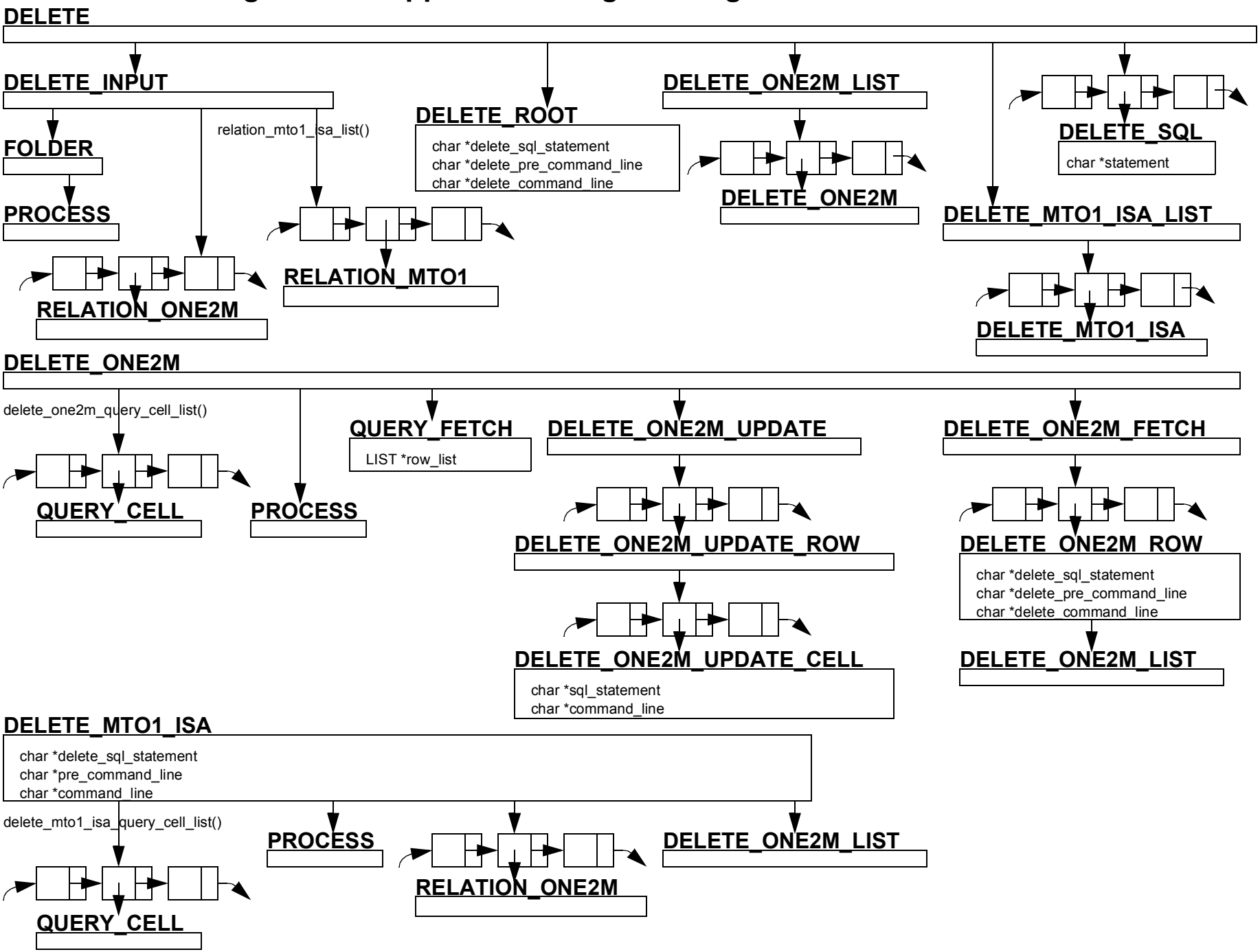
UPDATE_ATTRIBUTE

FOLDER_ATTRIBUTE *

DELETE_PRIMARY

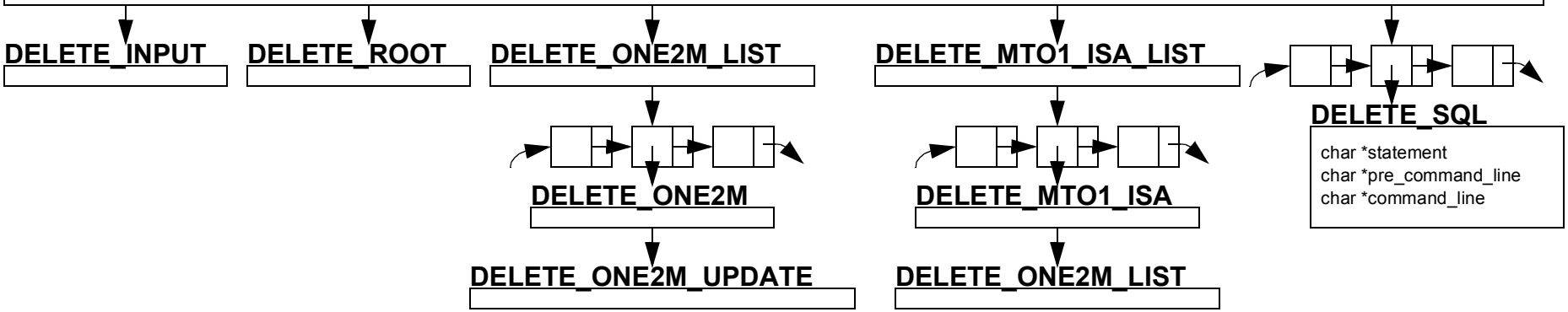
```
/* Usage */
DELETE_PRIMARY *delete_primary_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *primary_data_list_string,
    int row_number,
    int row_count,
    boolean delete_omit_isa_boolean );

/* Process */
DELETE_PRIMARY *delete_primary_calloc(
    void );

/* If error then exit( 1 ) */
SESSION *session_fetch(
    application_name,
    session_key,
    login_name,
    1 /* appaserver_user_boolean */ );

char *folder_name =
    security_sql_injection_escape(
        SECRUITY_ESCAPE_CHARACTER_STRING,
        folder_name );

LIST *folder_attribute_fetch_primary_key_list(
    this->folder_name );

LIST *delete_primary_query_cell_list(
    SQL_DELIMITER,
    primary_data_list_string,
    folder_attribute_fetch_primary_key_list() );

DELETE *delete =
    delete_new(
        application_name,
        login_name,
        role_name,
        folder_name,
        row_number,
        row_count,
        delete_primary_query_cell_list(),
        !delete_omit_isa_boolean
            /* isa_boolean (need to unwind) */,
        1 /* update_null_boolean */ );

/* Usage */
LIST *delete_primary_query_cell_list(
    char SQL_DELIMITER,
    char *primary_data_list_string,
    LIST *folder_attribute_fetch_primary_key_list() );

/* Process */
LIST *list = list_new();

list_rewind( folder_attribute_fetch_primary_key_list );

for (     p = 0;
          piece(  datum[],
                  SQL_DELIMITER,
                  primary_data_list_string,
                  p );
          p++ )
{
    QUERY_CELL *query_cell_parse(
        (LIST *)0 /* folder_attribute_list */,
        0 /* date_convert_format_enum */,
        list_get( folder_attribute_fetch_primary_key_list )
            /* attribute_name */,
        strdup( datum ) );

    query_cell_parse()->primary_key_index = p + 1;

    list_set(
        list,
        query_cell_parse() );

    list_next( folder_attribute_fetch_primary_key_list );
}

/* Driver */

/* Returns row_count */
int delete_execute(
    SQL_EXECUTABLE,
    delete_primary->
        delete->
        delete_input->
        appaserver_error_filename,
    delete_primary->delete->sql_statement_list,
    delete_primary->delete->pre_command_line_list,
    delete_primary->delete->command_line_list );

/* Attributes */
SESSION *session_fetch;
char *folder_name;
LIST *folder_attribute_fetch_primary_key_list;
LIST *query_cell_list;
DELETE *delete;
```




DELETE
DELETE_INPUT
FOLDER
PROCESS
RELATION_ONE2M
relation_mto1_isa_list()
RELATION_MTO1
DELETE_ROOT
char *delete_sql_statement
char *delete_pre_command_line
char *delete_command_line
DELETE_ONE2M_LIST
DELETE_ONE2M
DELETE_SQL
char *statement
DELETE_MTO1_ISA_LIST
DELETE_MTO1_ISA
DELETE_ONE2M
delete_one2m_query_cell_list()
QUERY_CELL
PROCESS
QUERY_FETCH
LIST *row_list
DELETE_ONE2M_UPDATE
DELETE_ONE2M_UPDATE_ROW
DELETE_ONE2M_UPDATE_CELL
char *sql_statement
char *command_line
DELETE_ONE2M_FETCH
DELETE_ONE2M_ROW
char *delete_sql_statement
char *delete_pre_command_line
char *delete_command_line
DELETE_ONE2M_LIST
DELETE_MTO1_ISA
char *delete_sql_statement
char *pre_command_line
char *command_line
delete_mto1_isa_query_cell_list()
QUERY_CELL
PROCESS
RELATION_ONE2M
DELETE_ONE2M_LIST

DELETE (1)

```
/* Usage */
DELETE *delete_new(
    char *application_name,
    char *login_name,
    char *role_name,
    char *folder_name,
    int row_number,
    int row_count,
    LIST *primary_query_cell_list,
    boolean isa_boolean,
    boolean update_null_boolean );

/* Process */
DELETE *delete_calloc( void );

DELETE_INPUT *delete_input =
    delete_input_new(
        application_name,
        login_name,
        role_name,
        folder_name );

if ( !delete_input->folder_operation_delete_boolean )
{
    return this;
}

DELETE_ROOT *delete_root =
    delete_root_new(
        application_name,
        folder_name,
        row_number,
        row_count,
        primary_query_cell_list,
        delete_input->folder->post_change_process,
        delete_input->security_entity_where,
        delete_input->appaserver_error_filename );

if ( list_length( delete_input->relation_one2m_list ) )
{
    DELETE_ONE2M_LIST *delete_one2m_list =
        delete_one2m_list_new(
            application_name,
            role_name,
            update_null_boolean,
            delete_input->
                appaserver_error_filename,
            delete_input->
                appaserver_parameter_mount_point,
            primary_query_cell_list,
            delete_input->relation_one2m_list );
}

if (    isa_boolean
&&  list_length(
            delete_input->
                relation_mto1_isa_list ) )
{
    DELETE_MTO1_ISA_LIST *delete_mto1_isa_list =
        delete_mto1_isa_list_new(
            application_name,
            role_name,
            delete_input->
                appaserver_error_flename,
            delete_input->
                appaserver_parameter_mount_point,
            primary_query_cell_list,
            delete_input->relation_mto1_isa_list );
}

LIST *delete_sql_list(
    delete_root,
    delete_one2m_list,
    delete_mto1_isa_list );

unsigned int delete_row_count(
    LIST *delete_sql_list );

unsigned int delete_update_cell_count(
    LIST *delete_sql_list );

char *delete_message_string(
    DELETE_MESSAGE,
    delete_row_count(),
    delete_update_cell_count() );

char *delete_complete_message_string =
    char *delete_message_string(
        DELETE_COMPLETE_MESSAGE,
        delete_row_count(),
        delete_update_cell_count() );

/* Attributes */
DELETE_INPUT *delete_input;
DELETE_ROOT *delete_root;
DELETE_ONE2M_LIST *delete_one2m_list;
DELETE_MTO1_ISA_LIST *delete_mto1_isa_list;
LIST *delete_sql_list;
unsigned int row_count;
unsigned int update_cell_count;
char *message_string;
char *complete_message_string;
```



DELETE (2)

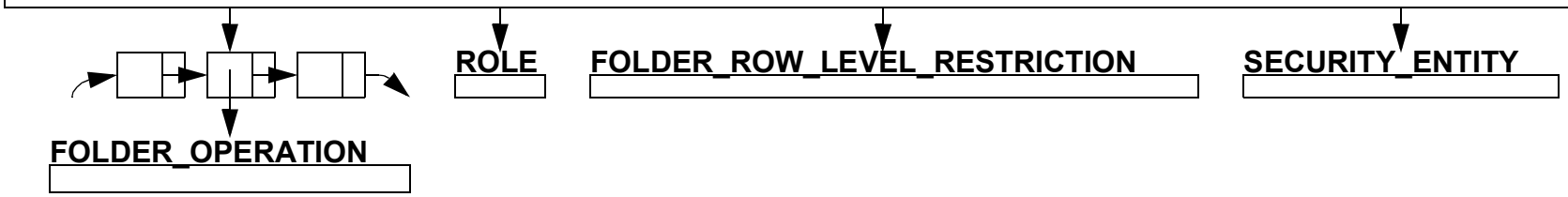
```
/* Usage */
char *delete_where(
    LIST *primary_query_cell_list,
    char *security_entity_where );

/* Process */
char *query_cell_where_string(
    primary_query_cell_list );

if ( security_entity_where )
{
    snprintf(
            static where[],
            sizeof ( where ),
            "%s and %s",
            query_cell_where_string(),
            security_entity_where );
}
else
{
    string_strcpy(
        static where[],
        query_cell_where_string(),
        sizeof ( where ) );
}

free( query_cell_where_string() );

/* Usage */
char *delete_sql_statement(
    char *appaserver_table_name(),
    char *delete_where() );

/* Process */
snprintf(
    sql_statement[],
    sizeof ( sql_statement ),
    "delete from %s where %s;",
    appaserver_table_name,
    delete_where );

/* Usage */
char *delete_command_line(
    const char *delete_state,
    int row_number,
    int row_count,
    LIST *primary_query_cell_list,
    char *post_change_process_command_line,
    char *appaserver_error_filespecification );

/* Process */
char *process_replace_command_line(
    UPDATE_PREUPDATE_PREFIX,
    (char *)0 /* application_name */,
    (char *)0 /* session_key */,
    (char *)0 /* login_name */,
    (char *)0 /* role_name */,
    (char *)0 /* folder_name */,
    (char *)0 /* target_frame */,
    delete_state,
    (char *)0 /* process_name */,
    (char *)0 /* many_folder_name */,
    (char *)0 /* one_folder_name */,
    (char *)0 /* related_column */,
    (char *)0 /* update_results_string */,
    (char *)0 /* update_error_string */,
    (DICTIONARY *)0 /* operation_row_list_dictionary */,
    (DICTIONARY *)0 /* dictionary_single_row */,
    (char *)0 /* where_string */,
    row_number,
    row_count,
    (pid_t)0 /* parent_process_id */,
    (LIST *)0 /* primary_key_data_list */,
    "y" /* execute_yn */,
    (LIST *)0 /* insert_datum_list */,
    (LIST *)0 /* update_attribute_list */,
    primary_query_cell_list,
    appaserver_error_filespecification,
    post_change_process_command_line
        /* input_command_line */ );

/* Usage */
LIST *delete_mto1_query_cell_list(
    LIST *primary_query_cell_list,
    LIST *one_folder_attribute_primary_key_list );

/* Process */
LIST *delete_query_cell_list(
    primary_query_cell_list,
    one_folder_attribute_primary_key_list );

/* Usage */
LIST *delete_one2m_query_cell_list(
    LIST *primary_query_cell_list,
    LIST *relation_foreign_key_list );

/* Process */
LIST *delete_query_cell_list(
    primary_query_cell_list,
    relation_foreign_key_list );

/* Usage */
LIST *delete_query_cell_list(
    LIST *primary_query_cell_list,
    LIST *key_list );

/* Process */
if (    list_length( primary_query_cell_list ) !=
        list_length( key_list ) )
{
    exit( 1 );
}

list_rewind( key_list );

for QUERY_CELL *query_cell in primary_query_cell_list
{
    list_set(
        cell_list,
        QUERY_CELL *query_cell_parse(
            (LIST *)0 /* folder_attribute_list */,
            0 /* destination_enum */,
            list_get( key_list ) /* attribute_name */,
            query_cell->select_datum ) );

    list_next( key_list );
}

/* Usage */
char *delete_message_string(
    const char *message,
    unsigned int delete_row_count,
    unsigned int delete_update_cell_count );

/* Process */
```

DELETE (3)

```
/* Usage */
char *delete_execute_system_string(
    const char *SQL_EXECUTABLE );

/* Process */
snprintf(
    static system_string[],
    sizeof ( system_string ),
    "tee_appaserver.sh | "
    "%s 2>&1 | "
    "html_paragraph_wrapper.e",
    SQL_EXECUTABLE );

/* Usage */
char *delete_key_list_data_string_sql(
    const char SQL_DELIMITER,
    char *table_name,
    LIST *key_list,
    char *data_string );

/* Process */
char sql[];
char *ptr = sql;
int p = 0;

ptr +=
    sprintf(
        ptr,
        "delete from %s where ",
        table_name );

for char *key in key_list
{
    if ( p ) ptr += sprintf( ptr, " and " );

    piece(  char datum[],
            SQL_DELIMITER,
            data_string,
            p );

    char *security_sql_injection_escape(
        SECURITY_ESCAPE_CHARACTER_STRING,
        datum );

    ptr +=
        sprintf(
            ptr,
            "%s = '%s'",
            key,
            security_sql_injection_escape() );

    free( security_sql_injection_escape() );
    p++;
}

ptr += sprintf( ptr, ";" );

/* Driver */
void delete_display(
    LIST *delete->delete_sql_list );

/* Process */
FILE *appaserver_output_pipe(
    "html_table.e '' sql_statement" );

for DELETE_SQL *delete_sql in delete_sql_list
{
    fprintf(
    appaserver_ouput_pipe(),
    "%s\n",
    delete_sql->statement );
}
pclose( appaserver_output_pipe() );

/* Driver */
int delete_execute(
    const char *SQL_EXECUTABLE,
    LIST *delete->delete_sql_list );

/* Process */
for DELETE_SQL *delete_sql in delete_sql_list
{
    if ( delete_sql->pre_command_line )
    {
        security_system(
            SECURITY_FORK_CHARACTER,
            SECURITY_FORK_STRING,
            delete_sql->pre_command_line );
    }
}

char *delete_execute_system_string(
    SQL_EXECUTABLE );

FILE *appaserver_output_pipe(
    delete_execute_system_string() );

for DELETE_SQL *delete_sql in delete_sql_list
{
    fprintf(
        appaserver_output_pipe(),
        "%s\n",
        delete_sql->statement );
}
pclose( appaserver_output_pipe() );

for DELETE_SQL *delete_sql in delete_sql_list
{
    if ( delete_sql->command_line )
    {
        security_system(
            SECURITY_FORK_CHARACTER,
            SECURITY_FORK_STRING,
            delete_sql->command_line );
    }
}
```

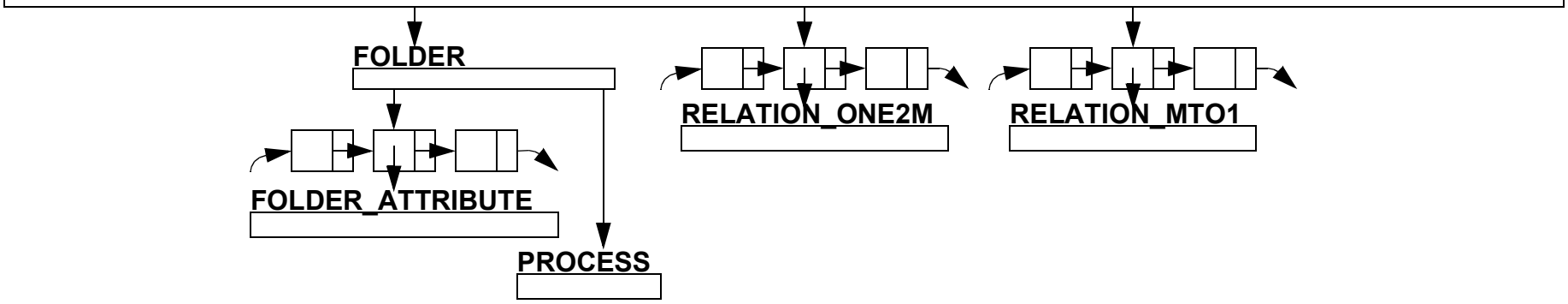
DELETE_INPUT (1)

```
/* Usage */
DELETE_INPUT *delete_input_new(
    char *application_name,
    char *login_name,
    char *role_name,
    char *folder_name );

/* Process */
DELETE_INPUT *delete_input_calloc( void );

LIST *folder_operation_list(
    folder_name,
    role_name,
    0 /* not fetch_operation */,
    0 /* not fetch_process */ );

boolean folder_operation_delete_boolean(
    folder_operation_list() );

if ( !folder_operation_delete_boolean() ) return this;

char *appaserver_error_filespecification(
    application_name );

char *appaserver_parameter_mount_point();

ROLE *role =
    role_fetch(
        role_name,
        0 /* not fetch_role_attribute_exclude_list */ );

FOLDER_ROW_LEVEL_RESTRICTION *
    folder_row_level_restriction =
        folder_row_level_restriction_fetch(
            folder_name );

SECURITY_ENTITY *security_entity =
    security_entity_new(
        login_name,
        folder_name,
        folder_row_level_restriction->
            non_owner_forbid ||
        folder_row_level_restriction->
            non_owner_viewonly,
        role->override_row_restrictions );
```



DELETE_INPUT (2)

```
/* Process (continued) */
FOLDER *folder =
    folder_fetch(
        folder_name,
        (LIST *)0 /* exclude_attribute_name_list */,
        /* Sets folder_attribute_primary_list */
        /* Sets folder_attribute_primary_key_list */
        /* Sets folder_attribute_name_list */
        1 /* fetch_folder_attribute_list */,
        1 /* fetch_attribute */,
        1 /* cache_boolean */ );

LIST *relation_one2m_list(
    folder_name
        /* one_folder_name */,
    folder->folder_attribute_primary_key_list
        /* one_folder_primary_key_list */,
    1 /* include_isa_boolean */ );

LIST *relation_mto1_isa_list(
    (LIST *)0 /* mto1_isa_list Pass in null */,
    folder_name
        /* many_folder_name */,
    folder->folder_attribute_primary_key_list
        /* many_folder_primary_key_list */,
    0 /* not fetch_relation_one2m_list */,
    0 /* not fetch_relation_mto1_list */ );

if ( folder->post_change_process_name )
{
    PROCESS *folder->post_change_process =
        process_fetch(
            folder->post_change_process_name,
            (char *)0 /* document_root */,
            (char *)0 /* relative_source */,
             1 /* check_executable_inside_filesystem */,
                appaserver_parameter_mount_point() );
}

/* Attributes */
LIST *folder_operation_list;
boolean folder_operation_delete_boolean;
char *appaserver_error_filespecification;
char *appaserver_parameter_mount_point;
ROLE *role;
FOLDER_ROW_LEVEL_RESTRICTION *
    folder_row_level_restriction;
SECURITY_ENTITY *security_entity;
FOLDER *folder;
char *security_entity_where;
LIST *relation_one2m_list;
LIST *relation_mto1_isa_list;
```



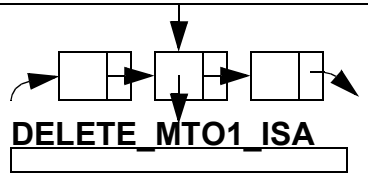
DELETE_ROOT

```
/* Usage */
DELETE_ROOT *delete_root_new(
    char *application_name,
    char *folder_name,
    int row_number,
    int row_count,
    LIST *primary_query_cell_list,
    PROCESS *post_change_process,
    char *security_entity_where,
    char *appaserver_error_filespecification );

/* Process */
DELETE_ROOT *delete_root_calloc(
    void );

char *appaserver_table_name(
    folder_name );

char *delete_where(
    primary_query_cell_list,
    security_entity_where );

char *delete_where() =
    strdup( delete_where() );

char *delete_sql_statement(
    appaserver_table_name(),
    delete_where() );

if ( post_change_process )
{
    char *delete_pre_command_line =
        char *delete_command_line(
            APPASERVER_PREDELETE_STATE,
            row_number,
            row_count,
            primary_query_cell_list,
            post_change_process->command_line,
            appaserver_error_filename );

    char *delete_command_line =
        char *delete_command_line(
            APPASERVER_DELETE_STATE,
            row_number,
            row_count,
            primary_query_cell_list,
            post_change_process->command_line,
            appaserver_error_filespecification );
}

/* Attributes */
char *delete_where;
char *delete_sql_statement;
char *delete_pre_command_line;
char *delete_command_line;
```

DELETE_MTO1_ISA_LIST

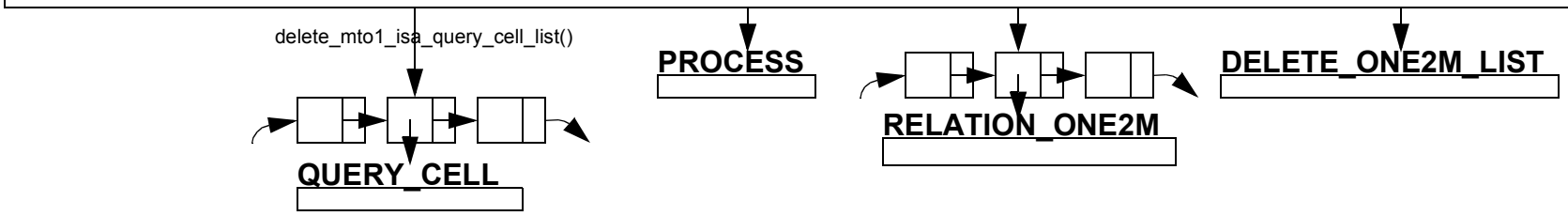
```
/* Usage */
DELETE_MTO1_ISA_LIST *
    delete_mto1_isa_list_new(
        char *application_name,
        char *role_name,
        boolean update_null_boolean,
        char *appaserver_error_filename,
        char *appaserver_parameter_mount_point,
        LIST *primary_query_cell_list,
        LIST *relation_mto1_isa_list );

/* Process */
DELETE_MTO1_ISA_LIST *delete_mto1_isa_list_calloc(
    void );
```

```
LIST *list = list_new();

for RELATION_MTO1 *relation_mto1_isa in
    relation_mto1_isa_list
{
    DELETE_MTO1_ISA *
        delete_mto1_isa =
            delete_mto1_isa_new(
                application_name,
                role_name,
                update_null_boolean,
                appaserver_error_flename,
                appaserver_parameter_mount_point,
                primary_query_cell_list,
                relation_mto1_isa->one_folder_name,
                relation_mto1_isa->
                    one_folder->
                    folder_attribute_primary_key_list,
                relation_mto1_isa->
                    one_folder->
                    post_change_process_name );

    list_set( list, delete_mto1_isa );
}

/* Attributes */
LIST *list;
```



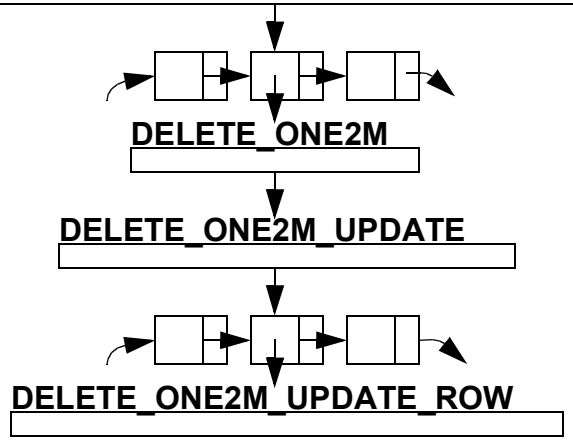
DELETE_MTO1_ISA

```
/* Usage */
DELETE_MTO1_ISA *delete_mto1_isa_new(
    char *application_name,
    char *role_name,
    boolean update_null_boolean,
    char *appaserver_error_filename(),
    char *appaserver_parameter_mount_point(),
    LIST *primary_query_cell_list,
    char *one_folder_name,
    LIST *one_folder_attribute_primary_key_list,
    char *post_change_process_name );

/* Process */
DELETE_MTO1_ISA *delete_mto1_isa_calloc(
    void );

char *appaserver_table_name(
    one_folder_name ),

LIST *delete_mto1_isa_query_cell_list(
    primary_query_cell_list,
    one_folder_attribute_primary_key_list );

char *delete_where(
    delete_mto1_isa_query_cell_list(),
    (char *)0 /* security_entity_where */ );

char *delete_sql_statement(
    appaserver_table_name(),
    delete_where() );

if ( post_change_process_name )
{
    PROCESS *post_change_process =
        process_fetch(
            post_change_process_name,
            (char *)0 /* document_root */,
            (char *)0 /* relative_source */,
            1 /* check_executable_inside_filesystem */,
            appaserver_parameter_mount_point );

    char *pre_command_line =
        delete_command_line(
            APPASERVER_PREDELETE_STATE,
            1 /* row_number */,
            1 /* row_count */,
            delete_mto1_isa_query_cell_list(),
            post_change_process->
                command_line,
            appaserver_error_filename );

    char *command_line =
        delete_command_line(
            APPASERVER_DELETE_STATE,
            1 /* row_number */,
            1 /* row_count */,
            delete_mto1_isa_query_cell_list(),
            post_change_process->
                command_line,
            appaserver_error_filename );
}

LIST *relation_one2m_list(
    one_folder_name,
    one_folder_attribute_primary_key_list,
    1 /* include_isa_boolean */ );

if ( list_length( relation_one2m_list() ) )
{
    DELETE_ONE2M_LIST *
        delete_one2m_list =
            delete_one2m_list_new(
                application_name,
                role_name,
                update_null_boolean,
                appaserver_error_filename,
                appaserver_parameter_mount_point,
                delete_mto1_query_cell_list(),
                relation_one2m_list()  );
}

/* Usage */
LIST *delete_mto1_isa_query_cell_list(
    LIST *primary_query_cell_list,
    LIST *one_folder_attribute_primary_key_list );

/* Process */
LIST *delete_query_cell_list(
    primary_query_cell_list,
    one_folder_attribute_primary_key_list );

/* Attributes */
char *one_folder_name;
LIST *query_cell_list;
char *delete_where;
char *delete_sql_statement;
PROCESS *post_change_process;
char *pre_command_line;
char *command_line;
LIST *relation_one2m_list;
DELETE_ONE2M_LIST *delete_one2m_list;
```



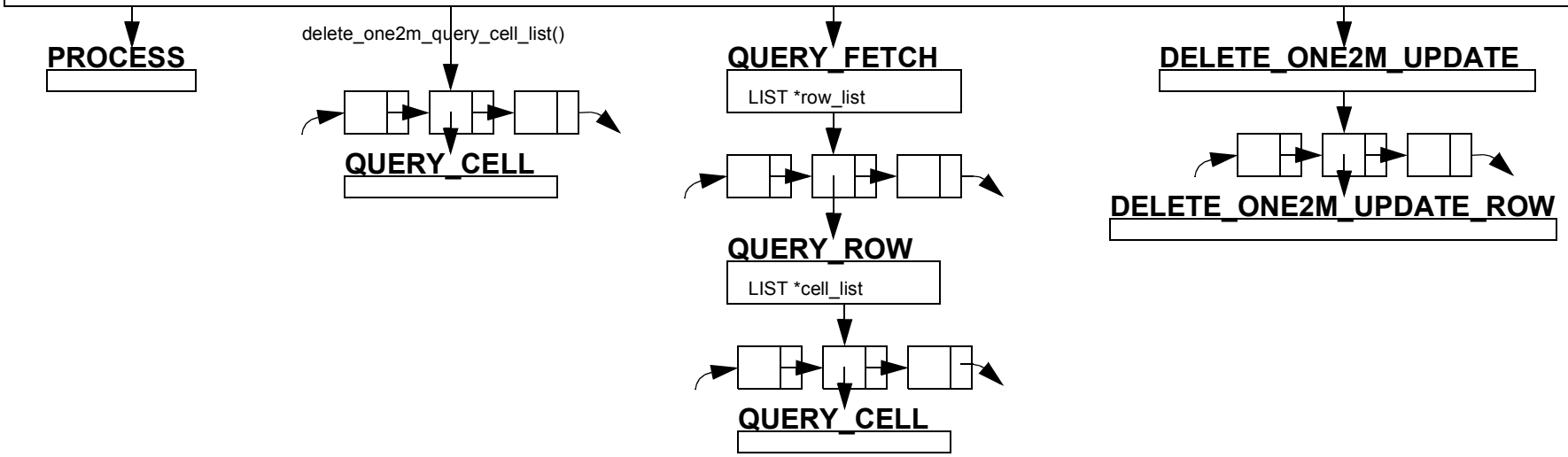
DELETE_ONE2M_LIST

```
/* Usage */
DELETE_ONE2M_LIST *delete_one2m_list_new(
    char *application_name,
    char *role_name,
    boolean update_null_boolean,
    char *appaserver_error_filename(),
    char *appaserver_parameter_mount_point(),
    LIST *primary_query_cell_list,
    LIST *relation_one2m_list() );

/* Process */
DELETE_ONE2M_LIST *delete_one2m_list_calloc(
    void );

for RELATION_ONE2M *relation_one2m in
    relation_one2m_list
{
    /* Cache PROCESS *post_change_process */
    if ( relation_one2m->
            many_folder->
            post_change_process_name
    && !relation_one2m->
            many_folder->
            post_change_process )
    {
        relation_one2m->
            many_folder->
            post_change_process =
                PROCESS *process_fetch(
                    relation_one2m->
                        many_folder->
                        post_change_process_name,
                    (char *)0 /* document_root */,
                    (char *)0 /* relative_source */,
                    1 /* check_inside */,
                    appaserver_parameter_mount_point );
    }

    DELETE_ONE2M *delete_one2m =
        delete_one2m_new(
            application_name,
            role_name,
            update_null_boolean,
            appaserver_error_filename,
            appaserver_parameter_mount_point,
            primary_query_cell_list
                /* one_folder_query_cell_list */,
            relation_one2m->many_folder_name,
            relation_one2m->
                many_folder->
                folder_attribute_primary_list
                /* many_folder_attribute_primary_list */,
            relation_one2m->
                many_folder->
                folder_attribute_primary_key_list
                /* many_folder_primary_key_list */,
            relation_one2m->relation_foreign_key_list,
            relation_one2m->foreign_key_none_primary,
            relation_one2m->
                many_folder->
                post_change_process );

        if ( delete_one2m )
        {
            if ( !list ) list = list_new();

            list_set(
                list,
                delete_one2m );
        }
    }

    /* Attribute */
    LIST *list;
```



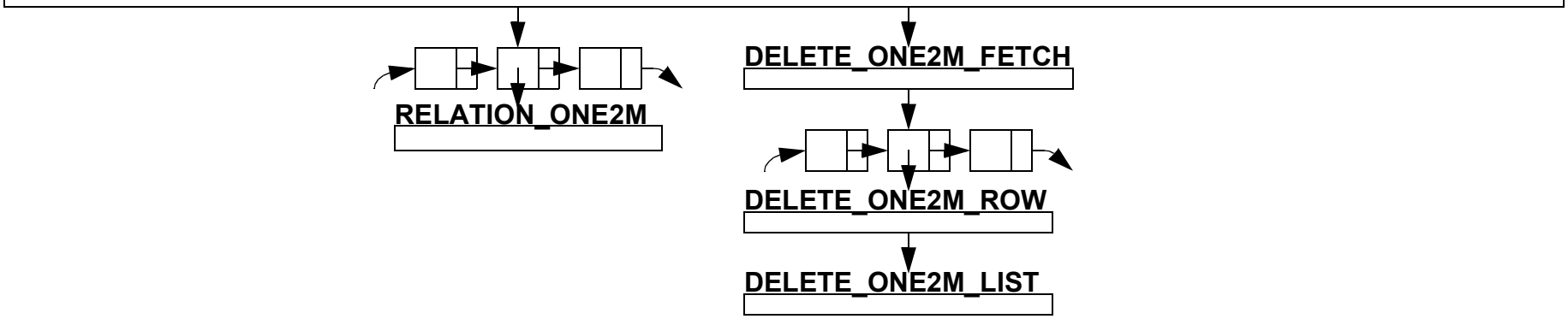
DELETE_ONE2M (1)

```
/* Usage */
DELETE_ONE2M *delete_one2m_new(
    char *application_name,
    char *role_name,
    boolean update_null_boolean,
    char *appaserver_error_filespecification,
    char *appaserver_parameter_mount_point,
    LIST *one_folder_query_cell_list,
    char *many_folder_name,
    LIST *many_folder_attribute_primary_list,
    LIST *many_folder_primary_key_list,
    LIST *relation_foreign_key_list,
    boolean foreign_key_none_primary,
    PROCESS *post_change_process );

/* Process */
DELETE_ONE2M *delete_one2m_calloc(
    void );

char *delete_one2m_table_name(
    char *application_name,
    char *many_folder_name );

LIST *delete_one2m_query_cell_list(
    one_folder_query_cell_list,
    relation_foreign_key_list );

char *delete_one2m_select_string(
    LIST *many_folder_primary_key_list,
    char delimiter /* ',' */ );

char *query_cell_where_string(
    delete_one2m_query_cell_list() );

char *query_system_string(
    application_name,
    delete_one2m_select_string(),
    many_folder_name /* from_string */,
    query_cell_where_string(),
    (char *)0 /* group_by_string */,
    (char *)0 /* order_string */,
    0 /* max_rows */ );

QUERY_FETCH *query_fetch =
    query_fetch_new(
        many_folder_name,
        many_folder_primary_attribute_list
            /* folder_attribute_list */,
        0 /* destination_enum */,
        many_folder_primary_key_list
            /* select_name_list */,
        query_system_string(),
            1 /* input_save_boolean for MERGE_PURGE */ );

int delete_one2m_row_count(
    query_fetch->row_list );

if ( !delete_one2m_row_count() ) return NULL;

if ( foreign_key_none_primary )
{
    if ( update_null_boolean )
    {
        DELETE_ONE2M_UPDATE *
            delete_one2m_update =
                delete_one2m_update_new(
                    appaserver_error_filespecification,
                    many_folder_name,
                    delete_one2m_table_name(),
                    delete_one2m_query_cell_list()
                        /* set_null_query_cell_list */,
                    post_change_process,
                    query_fetch->row_list,
                    delete_one2m_row_count() );
    }
    return this;
}
```



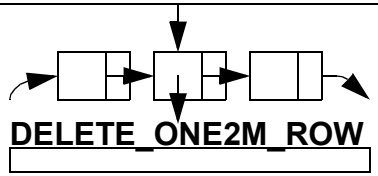
DELETE_ONE2M (2)

```
/* Process (continued) */
LIST *relation_one2m_list(
    many_folder_name
        /* one_folder_name */,
    many_folder_primary_key_list
        /* one_folder_primary_key_list */,
    1 /* include_isa_boolean */ );

DELETE_ONE2M_FETCH *
    delete_one2m_fetch =
        delete_one2m_fetch_new(
            application_name,
            role_name,
            update_null_boolean,
            appaserver_error_filename,
            appaserver_parameter_mount_point,
            delete_one2m_table_name()
                /* many_appaserver_table_name */,
            post_change_process,
            query_fetch->row_list,
            relation_one2m_list() );

/* Usage */
LIST *delete_one2m_query_fetch_list(
    LIST *delete_one2m_list );

/* Process */
LIST *list = list_new();

for DELETE_ONE2M *delete_one2m in
    delete_one2m_list
{
    list_set(
        list,
        delete_one2m->query_fetch );
}

/* Attributes */
PROCESS *post_change_process;
char *many_folder_name;
char *table_name;
LIST *delete_one2m_query_cell_list;
char *select_string;
char *query_cell_where_string;
char *query_system_string;
QUERY_FETCH *query_fetch;
int row_count;
DELETE_ONE2M_UPDATE *delete_one2m_update;
LIST *relation_one2m_list;
DELETE_ONE2M_FETCH *delete_one2m_fetch;
```



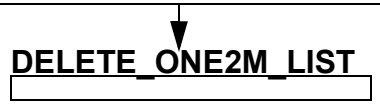
DELETE_ONE2M_FETCH

```
/* Usage */
DELETE_ONE2M_FETCH *
   delete_one2m_fetch_new(
      char *application_name,
      char *role_name,
      boolean update_null_boolean,
      char *appaserver_error_filename,
      char *appaserver_parameter_mount_point,
      char *many_appaserver_table_name,
      PROCESS *post_change_process,
      LIST *query_fetch_row_list,
      LIST *relation_one2m_list() );

/* Process */
DELETE_ONE2M_FETCH *
   delete_one2m_fetch_calloc(
      void );

LIST *delete_one2m_row_list = list_new();

for QUERY_ROW *query_row in query_fetch_row_list
{
   DELETE_ONE2M_ROW *
      delete_one2m_row =
         delete_one2m_row_new(
            application_name,
            role_name,
            appaserver_error_filename,
            appaserver_parameter_mount_point,
            many_appaserver_table_name,
            post_change_process,
            query_row->cell_list,
            relation_one2m_list );

   list_set(
      delete_one2m_row_list,
      delete_one2m_row );
}

/* Attribute */
LIST *delete_one2m_row_list;
```



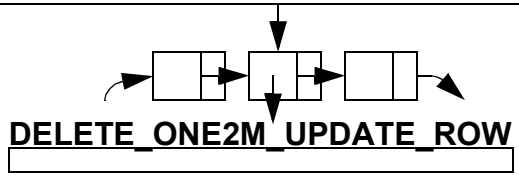
DELETE_ONE2M_ROW

```
/* Usage */
DELETE_ONE2M_ROW *
    delete_one2m_row_new(
        char *application_name,
        char *role_name,
        boolean update_null_boolean,
        char *appaserver_error_filename,
        char *appaserver_parameter_mount_point,
        char *many_appaserver_table_name,
        PROCESS *post_change_process,
        LIST *query_row_cell_list,
        LIST *relation_one2m_list() );

/* Process */
DELETE_ONE2M_ROW *
    delete_one2m_row_calloc(
        void );

char *delete_where(
    query_row_cell_list
        /* primary_query_cell_list */,
    (char *)0 /* security_entity_where */ );

char *delete_sql_statement(
    many_appaserver_table_name,
    delete_where() );

if ( post_change_process )
{
    char *pre_command_line =
        char *delete_command_line(
            APPASERVER_PREDELETE_STATE,
            1 /* row_number */,
            1 /* row_count */,
            query_row_cell_list
                /* primary_query_cell_list */,
            post_change_process->command_line,
            appaserver_error_filename );

    char *command_line =
        char *delete_command_line(
            APPASERVER_DELETE_STATE,
            1 /* row_number */,
            1 /* row_count */,
            query_row_cell_list
                /* primary_query_cell_list */,
            post_change_process->command_line,
            appaserver_error_filename );
}

if ( list_length( relation_one2m_list ) )
{
    DELETE_ONE2M_LIST *
        delete_one2m_list =
            delete_one2m_list_new(
                application_name,
                role_name,
                update_null_boolean,
                appaserver_error_filename,
                appaserver_parameter_mount_point,
                query_row_cell_list
                    /* primary_query_cell_list */,
                relation_one2m_list );
}

/* Attributes */
char *delete_where;
char *delete_sql_statement;
char *pre_command_line;
char *command_line;
DELETE_ONE2M_LIST *delete_one2m_list;
```



DELETE_ONE2M_UPDATE

```
/* Usage */
DELETE_ONE2M_UPDATE *
    delete_one2m_update_new(
        char *appaserver_error_filespecification,
        char *folder_name,
        char *appaserver_table_name,
        LIST *set_null_query_cell_list,
        PROCESS *post_change_process,
        LIST *query_fetch_row_list,
        int delete_one2m_row_count );

/* Process */
DELETE_ONE2M_UPDATE *
    delete_one2m_update_calloc(
        void );

LIST *delete_one2m_update_row_list = list_new();

for QUERY_ROW *query_row in query_fetch_row_list
{
    DELETE_ONE2M_UPDATE_ROW *
        delete_one2m_update_row =
            delete_one2m_update_row_new(
                appaserver_error_filespecification,
                appaserver_table_name,
                set_null_query_cell_list,
                post_change_process,
                ++row_number,
                delete_one2m_row_count,
                query_row->cell_list );

    list_set(
        delete_one2m_update_row_list,
        delete_one2m_update_row );
}

/* Attribute */
char *folder_name;
LIST *delete_one2m_update_row_list;
```

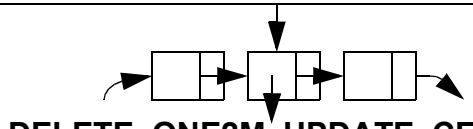


DELETE_ONE2M_UPDATE_ROW

DELETE_ONE2M_UPDATE_ROW

```
/* Usage */
DELETE_ONE2M_UPDATE_ROW *
   delete_one2m_update_row_new(
      char *appaserver_error_filespecification,
      char *appaserver_table_name,
      LIST *set_null_query_cell_list,
      PROCESS *post_change_process,
      int row_number,
      int row_count,
      LIST *query_row_cell_list );

/* Process */
DELETE_ONE2M_UPDATE_ROW *
   delete_one2m_update_row_calloc(
      void );
```

```
char *query_cell_where_string(
   query_row_cell_list );

LIST *delete_one2m_update_cell_list = list_new();

for QUERY_CELL *set_null_query_cell in
   set_null_query_cell_list
{
   DELETE_ONE2M_UPDATE_CELL *
      delete_one2m_update_cell =
         delete_one2m_update_cell_new(
            APPASERVER_UPDATE_STATE,
            UPDATE_PREUPDATE_PREFIX,
            appaserver_error_filespecification,
            appaserver_table_name,
            post_change_process,
            row_number,
            row_count,
            query_row_cell_list,
            query_cell_where_string(),
            set_null_query_cell );

   list_set(
      delete_one2m_update_cell_list,
      delete_one2m_update_cell );
}

free( query_cell_where_string() );

/* Attribute */
LIST *delete_one2m_update_cell_list;
```



DELETE_ONE2M_UPDATE_CELL

```
char *sql_statement
char *command_line
```

DELETE_ONE2M_UPDATE_CELL

```
/* Usage */
DELETE_ONE2M_UPDATE_CELL *
    delete_one2m_update_cell_new(
        char *APPASERVER_UPDATE_STATE,
        char *UPDATE_PREUPDATE_PREFIX,
        char *appaserver_error_filespecification,
        char *appaserver_table_name,
        PROCESS *post_change_process,
        int row_number,
        int row_count,
        LIST *query_row_cell_list,
        char *query_cell_where_string(),
        QUERY_CELL *set_null_query_cell );

/* Process */
DELETE_ONE2M_UPDATE_CELL *
    delete_one2m_update_cell_calloc(
        void );

char *delete_one2m_update_cell_sql_statement(
    appaserver_table_name,
    set_null_query_cell->attribute_name
        /* set_null_attribute_name */,
    query_cell_where_string );

if ( post_change_process )
{
    char *delete_one2m_update_cell_command_line(
        APPASERVER_UPDATE_STATE,
        UPDATE_PREUPDATE_PREFIX,
        appaserver_error_filename,
        set_null_query_cell->attribute_name,
        set_null_query_cell->select_datum,
        row_number,
        row_count,
        query_row_cell_list,
        post_change_process->command_line );
}

/* Usage */
char *delete_one2m_update_cell_sql_statement(
    char *appaserver_table_name(),
    char *set_null_attribute_name,
    char *query_cell_where_string() );

/* Process */
snprintf(
    char sql_statement[],
    sizeof ( sql_statement ),
    "update %s set %s = null where %s;",
    appaserver_table_name,
    set_null_attribute_name,
    query_cell_where_string );

/* Usage */
char *delete_one2m_update_cell_command_line(
    const char *APPASERVER_UPDATE_STATE,
    const char *UPDATE_PREUPDATE_PREFIX,
    char *appaserver_error_filespecification,
    char *query_cell_attribute_name,
    char *query_cell_select_datum,
    int row_number,
    int row_count,
    LIST *query_row_cell_list,
    char *post_change_process_command_line );

/* Process */
LIST *delete_one2m_update_cell_update_attribute_list(
    query_cell_attribute_name,
    query_cell_select_datum );

char *process_replace_command_line(
    UPDATE_PREUPDATE_PREFIX,
    (char *)0 /* application_name */,
    (char *)0 /* session_key */,
    (char *)0  /* login_name */,
    (char *)0 /* role_name */,
    (char *)0 /* folder_name */,
    (char *)0 /* target_frame */,
    APPASERVER_UPDATE_STATE,
    (char *)0 /* process_name */,
    (char *)0 /* many_folder_name */,
    (char *)0 /* one_folder_name */,
    (char *)0 /* related_column */,
    (char *)0 /* update_results_string */,
    (char *)0 /* update_error_string */,
    (DICTIONARY *)0 /* operation_row_list_dictionary */,
    (DICTIONARY *)0 /* dictionary_single_row */,
    (char *)0 /* where_string */,
    row_number,
    row_count,
    0 /* parent_process_id */,
    (LIST *)0 /* primary_key_data_list */,
    "y" /* execute_yn */,
    (LIST *)0 /* insert_datum_list */,
    delete_one2m_update_cell_update_attribute_list(),
    query_row_cell_list /* query_cell_list */,
    appaserver_error_filespecification,
    post_change_process_command_line
        /* input_command_line */ );

/* Usage */
LIST *delete_one2m_update_cell_update_attribute_list(
    char *query_cell_attribute_name,
    char *query_cell_select_datum );

/* Process */
UPDATE_ATTRIBUTE *update_attribute_new(
    query_cell_attribute_name,
    query_cell_select_datum /* post_datum */,
    (char *)0 /* file_datum */ );

list_set( list, update_attribute_new() );

/* Attributes */
char *sql_statement;
char *command_line;
```

DELETE_SQL (1)

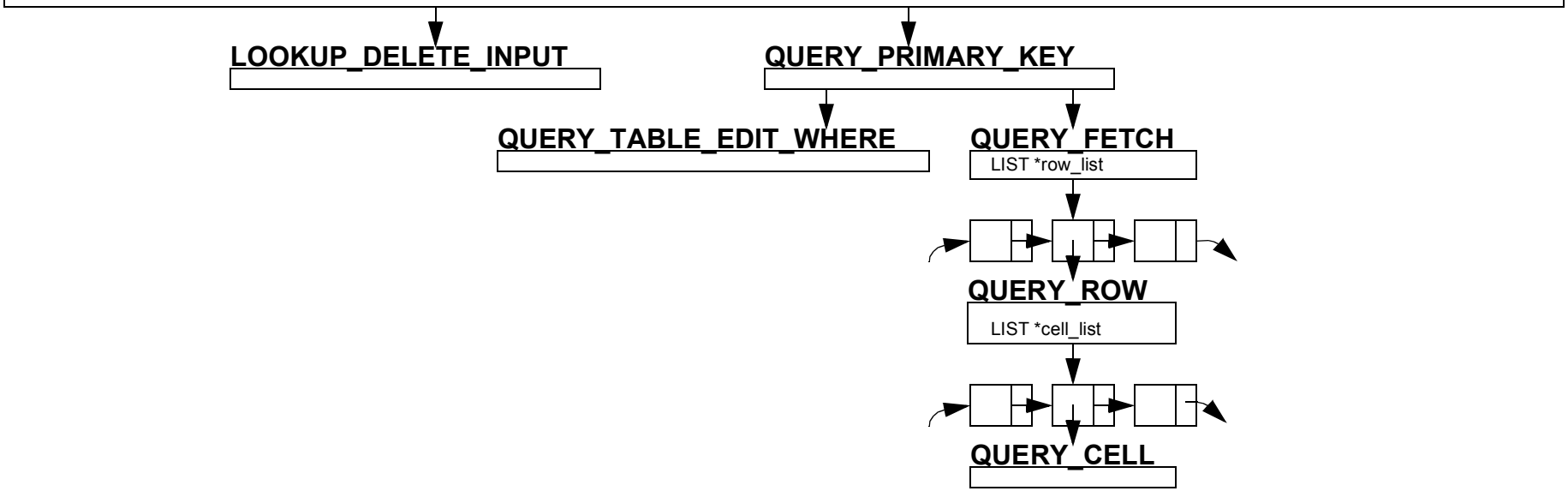
```
/* Usage */
LIST *delete_sql_list(
    DELETE_ROOT *delete_root,
    DELETE_ONE2M_LIST *delete_one2m_list,
    DELETE_MTO1_ISA_LIST *delete_mto1_isa_list );

/* Process */
LIST *sql_list = list_new();

DELETE_SQL *delete_sql_new(
    delete_root->delete_sql_statement,
    delete_root->delete_pre_command_line,
    delete_root->delete_command_line );

(void)delete_sql_getset(
    sql_list /* delete_sql_list in/out */,
    delete_sql_new() /* candidate_delete_sql */ );

if ( delete_one2m_list )
{
    delete_sql_one2m(
        sql_list /* delete_sql_list in/out */,
        delete_one2m_list->list
            /* delete_one2m_list */ );
}

if ( delete_mto1_isa_list )
{
    delete_sql_mto1_isa(
        sql_list /* delete_sql_list in/out */,
        delete_mto1_isa_list->list
            /* delete_mto1_isa_list */ );
}

/* Usage */
DELETE_SQL *delete_sql_new(
    char *statement,
    char *pre_command_line,
    char *command_line );

/* Process */
DELETE_SQL *delete_sql_calloc(
    void );

/* Usage */
boolean delete_sql_getset(
    LIST *delete_sql_list /* in/out */,
    DELETE_SQL *candidate_delete_sql );

/* Process */
for DELETE_SQL *delete_sql in delete_sql_list
{
    boolean delete_sql_duplicate_boolean(
        delete_sql,
        candidate_delete_sql );

    if ( delete_sql_duplicate_boolean() ) return 0;
}

list_set( delete_sql_list, candidate_delete_sql );

/* Usage */
boolean delete_sql_duplicate_boolean(
    DELETE_SQL *delete_sql,
    DELETE_SQL *candidate_delete_sql );

/* Process */
boolean duplicate_boolean = 0;

int strcmp(
    delete_sql->statement,
    candidate_delete_sql->statement );

if ( strcmp() == 0 ) duplicate_boolean = 1;

/* Usage */
void delete_sql_free(
    DELETE_SQL *delete_sql );

/* Process */
free( delete_sql );

/* Attributes */
char *statement;
char *pre_command_line;
char *command_line;
```

DELETE_SQL (2)

```
/* Usage */
void delete_sql_one2m(
    LIST *delete_sql_list /* in/out */,
    LIST *delete_one2m_list );

/* Process */
for DELETE_ONE2M *delete_one2m in
    delete_one2m_list
{
    void delete_sql_one2m_delete_update(
        delete_sql_list /* in/out */,
        delete_one2m->delete_one2m_fetch,
        delete_one2m->delete_one2m_update );
}

/* Usage */
void delete_sql_one2m_delete_update(
    LIST *delete_sql_list /* in/out */,
    DELETE_ONE2M_FETCH *
        delete_one2m_fetch,
    DELETE_ONE2M_UPDATE *
        delete_one2m_update );

/* Process */
if ( delete_one2m_fetch )
{
    delete_sql_one2m_delete(
        delete_sql_list /* in/out */,
        delete_one2m_fetch->
            delete_one2m_row_list );
}
else
{
    delete_sql_one2m_update(
        delete_sql_list /* in/out */,
        delete_one2m_update->
            delete_one2m_update_row_list );
}

/* Usage */
void delete_sql_one2m_delete(
    LIST *delete_sql_list /* in/out */,
    LIST *delete_one2m_row_list );

/* Process */
for DELETE_ONE2M_ROW *delete_one2m_row in
    delete_one2m_row_list
{
    DELETE_SQL *delete_sql_new(
        delete_one2m_row->delete_sql_statement,
        delete_one2m_row->pre_command_line,
        delete_one2m_row->command_line );

    boolean delete_sql_getset(
        delete_sql_list /* in/out */,
        delete_sql_new() /* candidate_delete_sql */ );

    if ( !delete_sql_getset() )
    {
        delete_sql_free( delete_sql_new() );
    }

    if ( delete_one2m_row->delete_one2m_list )
    {
        delete_sql_one2m(
            delete_sql_list /* in/out */,
            delete_one2m_row->
                delete_one2m_list->list
                    /* delete_one2m_list */ );
    }
}

/* Usage */
void delete_sql_one2m_update(
    LIST *delete_sql_list /* in/out */,
    LIST *delete_one2m_update_row_list );

/* Process */
for DELETE_ONE2M_UPDATE_ROW *
    delete_one2m_update_row in
        delete_one2m_update_row_list
{
    delete_sql_update_cell(
        delete_sql_list /* in/out */,
        delete_one2m_update_row->
            delete_one2m_update_cell_list );
}

/* Usage */
void delete_sql_update_cell(
    LIST *delete_sql_list /* in/out */,
    LIST *delete_one2m_update_cell_list );

/* Process */
for DELETE_ONE2M_UPDATE_CELL *
    delete_one2m_update_cell in
        delete_one2m_update_cell_list
{
    DELETE_SQL *delete_sql_new(
        delete_one2m_update_cell->sql_statement,
        (char *)0 /* pre_command_line */,
        delete_one2m_update_cell->command_line );

    boolean delete_sql_getset(
        delete_sql_list /* in/out */,
        delete_sql_new() /* candidate_delete_sql */ );

    if ( !delete_sql_getset() )
    {
        delete_sql_free( delete_sql_new() );
    }
}

/* Usage */
void delete_sql_mto1_isa(
    LIST *delete_sql_list /* in/out */,
    LIST *delete_mto1_isa_list );

/* Process */
for DELETE_MTO1_ISA *delete_mto1_isa in
    delete_mto1_isa_list
{
    DELETE_SQL *delete_sql_new(
        delete_mto1_isa->delete_sql_statement,
        delete_mto1_isa->pre_command_line,
        delete_mto1_isa->command_line );

    boolean delete_sql_getset(
        delete_sql_list /* in/out */,
        delete_sql_new() /* candidate_delete_sql */ );

    if ( !delete_sql_getset() )
    {
        delete_sql_free( delete_sql_new() );
    }

    if ( delete_mto1_isa->delete_one2m_list )
    {
        delete_sql_one2m(
            delete_sql_list /* in/out */,
            delete_mto1_isa->
                delete_one2m_list->list
                    /* delete_one2m_list */ );
    }
}
```

LOOKUP_DELETE (1)

```
/* Usage */
LOOKUP_DELETE *lookup_delete_new(
    char *session_folder->session->application_name,
    char *session_folder->session->login_name,
    char *session_folder->role_name,
    char *session_folder->folder_name,
    DICTIONARY *post_dictionary->
        original_post_dictionary );

/* Process */
LOOKUP_DELETE *lookup_delete_calloc(
    void );

LOOKUP_DELETE_INPUT *lookup_delete_input =
    lookup_delete_input_new(
        application_name,
        login_name,
        role_name,
        folder_name,
        DICTIONARY *original_post_dictionary );

if ( !lookup_delete_input->
        folder_operation_delete_boolean )
{
    exit( 1 );
}

QUERY_PRIMARY_KEY *query_primary_key =
    /* Note: frees each where. */
    query_primary_key_fetch(
        application_name,
        login_name,
        folder_name,
        lookup_delete_input->security_entity_where,
        lookup_delete_input->
            relation_mto1_list,
        lookup_delete_input->
            relation_mto1_isa_list,
        lookup_delete_input->
            folder_attribute_append_isa_list,
        lookup_delete_input->
            query_dictionary,
        (char *)0 /* additional_attribute_name */ );

char *lookup_delete_title_string(
    char *folder_name );

char *lookup_delete_sub_title_string(
    char *query_primary_key->
        query_table_edit_where->
        string );

int lookup_delete_row_count(
    int list_length(
        query_primary_key->
            query_fetch->
            row_list )
            /* query_fetch_row_list_length */ );

/* Attributes */
LOOKUP_DELETE_INPUT *lookup_delete_input;
QUERY_PRIMARY_KEY *query_primary_key;
char *title_string;
char *sub_title_string;
int row_count;
```



LOOKUP_DELETE (2)

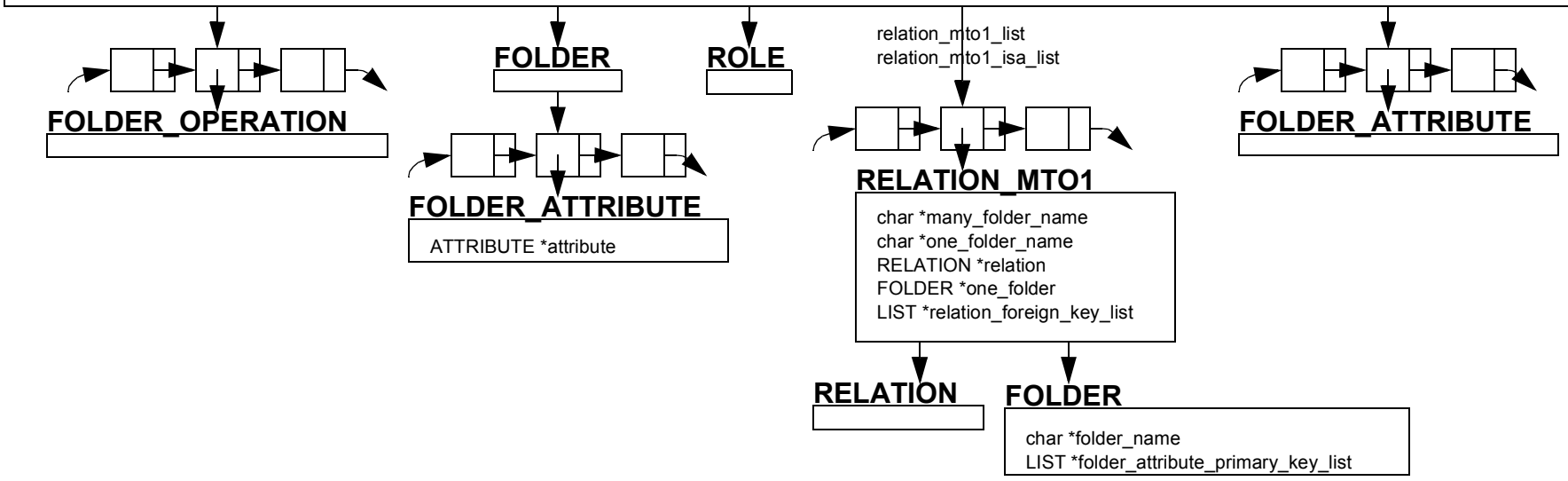
```
/* Driver */
void lookup_delete_state_one_form_output(
    const char *LOOKUP_DELETE_EXECUTABLE,
    char *session_folder->application_name,
    char *session_folder->session_key,
    char *session_folder->login_name,
    char *session_folder->role_name,
    char *session_folder->folder_name,
    DICTIONARY *post_dictionary->
        original_post_dictionary );

/* Process */
LOOKUP_DELETE *lookup_delete =
    lookup_delete_new(
        application_name,
        login_name,
        role_name,
        folder_name,
        original_post_dictionary );

void document_process_output(
    application_name,
    (LIST *)0 /* javascript_filename_list */,
    lookup_delete->title_string );

printf(
    "<h2>Where: %s</h2>\n",
    lookup_delete->sub_title_string );

if ( !lookup_delete->fetch_row_count )
{
    printf(   "<h3>%s</h3>\n",
              LOOKUP_DELETE_NO_ROWS_MESSAGE );
    void document_close();
    exit( 0 );
}

char *lookup_delete_prompt_html(
    char *folder_name,
    int lookup_delete->fetch_row_count );

printf(
    "%s\n",
    lookup_delete_prompt_html() );

char *lookup_delete_action_string(
    const char *LOOKUP_DELETE_EXECUTABLE,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name );

FORM_LOOKUP_DELETE *form_lookup_delete =
    form_lookup_delete_new(
        LOOKUP_DELETE_EXECUTE_LABEL,
        lookup_delete->
            lookup_delete_input->
            query_dictionary,
        lookup_delete_action_string() );

printf(
    "%s\n",
    form_lookup_delete->form_html );

void document_close();
```

LOOKUP_DELETE (3)

```
/* Driver */
void lookup_delete_state_two_execute(
    char *session_folder->application_name,
    char *session_folder->login_name,
    char *session_folder->role_name,
    char *session_folder->folder_name,
    boolean lookup_delete->
        lookup_delete_input->
        execute_boolean,
    LIST *query_fetch_row_list(),
    char *lookup_delete->title_string,
    int lookup_delete->row_count );

/* Process */
void document_process_output(
    application_name,
    (LIST *)0 /* javascript_filename_list */,
    title_string );

if ( list_rewind( query_fetch_row_list ) )
{
    for QUERY_ROW *query_row in
        query_fetch_row_list
    {

DELETE *delete =
    delete_new(
    application_name,
    login_name,
    role_name,
    folder_name,
    ++row_number,
    row_count,
    query_row->cell_list /* primary_cell_list */,
    1 /* isa_boolean */,
    1 /* update_null_boolean */ );

if ( !delete->folder_operation_delete ) exit( 1 );

if ( execute_boolean )
{
    void delete_execute(
        SQL_EXECUTABLE,
        delete->delete_sql_list );

    printf(   "<h3>%s</h3>\n",
              delete->complete_message_string );
}
else
    {
        void delete_display(
            delete->delete_sql_list );

        printf(   "<h3>%s</h3>\n",
                  delete->message_string );
    }

    } /* for each query_fetch_row_list */

    if ( !execute_boolean )
    {
        BUTTON *back_button();

        printf( "%s\n", back_button()->html );
    }

} /* if list_rewind() */
else
{
    printf(   "<h3>Nothing selected to delete.</h3>\n" );
}

void document_close();
```

LOOKUP_DELETE_INPUT (1)

```
/* Usage */
LOOKUP_DELETE_INPUT *
   lookup_delete_input_new(
      char *application_name,
      char *login_name,
      char *role_name,
      char *folder_name,
      DICTIONARY *original_post_dictionary );

/* Process */
LOOKUP_DELETE_INPUT *
   lookup_delete_input_calloc(
      void );

LIST *folder_operation_list(
    folder_name,
    role_name,
    0 /* not fetch_operation */,
    0 /* not fetch_process */ );

boolean folder_operation_delete_boolean(
   folder_operation_list() );

if ( !folder_operation_delete_boolean() )
   return this;

FOLDER *folder =
   folder_fetch(
      folder_name,
      (LIST *)0 /* exclude_attribute_name_list */,
      /* Sets folder_attribute_primary_list */
      /* Sets folder_attribute_primary_key_list */
      /* Sets folder_attribute_name_list */
      1 /* fetch_folder_attribute_list */,
      1 /* fetch_attribute */,
      1 /* cache_boolean */ );

ROLE *role =
   role_fetch(
      role_name,
      0 /* not fetch_role_attribute_exclude_list */ );

LIST *relation_mto1_list(
   folder_name
      /* many_folder_name */,
   folder->folder_attribute_primary_key_list
      /* many_primary_key_list */ );

LIST *relation_mto1_isa_list(
   (LIST *)0 /* mto1_isa_list Pass in null */,
   folder_name
      /* many_folder_name */,
   folder->folder_attribute_primary_key_list
      /* many_primary_key_list */,
   0 /* not fetch_relation_one2m_list */,
   1 /* not fetch_relation_mto1_list */ );

LIST *folder_attribute_append_isa_list(
   folder->folder_attribute_list /* append_isa_list */,
   relation_mto1_isa_list() );

LIST *folder_attribute_date_name_list(
   folder_attribute_append_isa_list() );
```



LOOKUP_DELETE_INPUT (2)

```
/* Process (continued) */
FOLDER_ROW_LEVEL_RESTRICTION *
   folder_row_level_restriction =
      folder_row_level_restriction_fetch(
         folder_name );

SECURITY_ENTITY *security_entity =
   security_entity_new(
      login_name,
      folder_name,
      folder_row_level_restriction->
         non_owner_forbid ||
      folder_row_level_restriction->
         non_owner_viewonly,
      role->override_row_restrictions );

DICTIONARY_SEPARATE_LOOKUP_DELETE *
   dictionary_separate =
      dictionary_separate_lookup_delete_new(
         original_post_dictionary,
         application_name,
         login_name,
         folder_attribute_date_name_list(),
         folder_attribute_append_isa_list() );

DICTIONARY *lookup_delete_input_query_dictionary(
   DICTIONARY *dictionary_separate->
      query_dictionary,
   DICTIONARY *dictionary_separate->
      non_prefixed_dictionary );

boolean lookup_delete_input_execute_boolean(
   char *LOOKUP_DELETE_EXECUTE_LABEL,
   DICTIONARY *dictionary_separate->
      non_prefixed_dictioinary );
```

```
/* Attribues */
LIST *folder_operation_list;
boolean folder_operation_delete_boolean;
FOLDER *folder;
ROLE *role;
LIST *relation_mto1_list;
LIST *relation_mto1_isa_list;
LIST *folder_attribute_append_isa_list;
LIST *folder_attribute_date_name_list;
FOLDER_ROW_LEVEL_RESTRICTION *
   folder_row_level_restriction;
SECURITY_ENTITY *security_entity;
DICTIONARY_SEPARATE_LOOKUP_DELETE *
   dictionary_separate;
DICTIONARY *query_dictionary;
boolean execute_boolean;
```

FOLDER_ROW_LEVEL_RESTRICTION

SECURITY_ENTITY

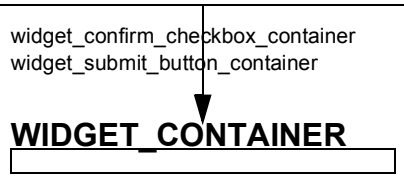
DICTIONARY

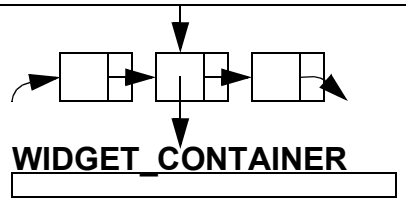
DICTIONARY_SEPARATE_LOOKUP_DELETE

```
DICTIONARY *query_dictionary
DICTIONARY *non_prefixed_dictionary
DICTIONARY *drillthru_dictionary
```

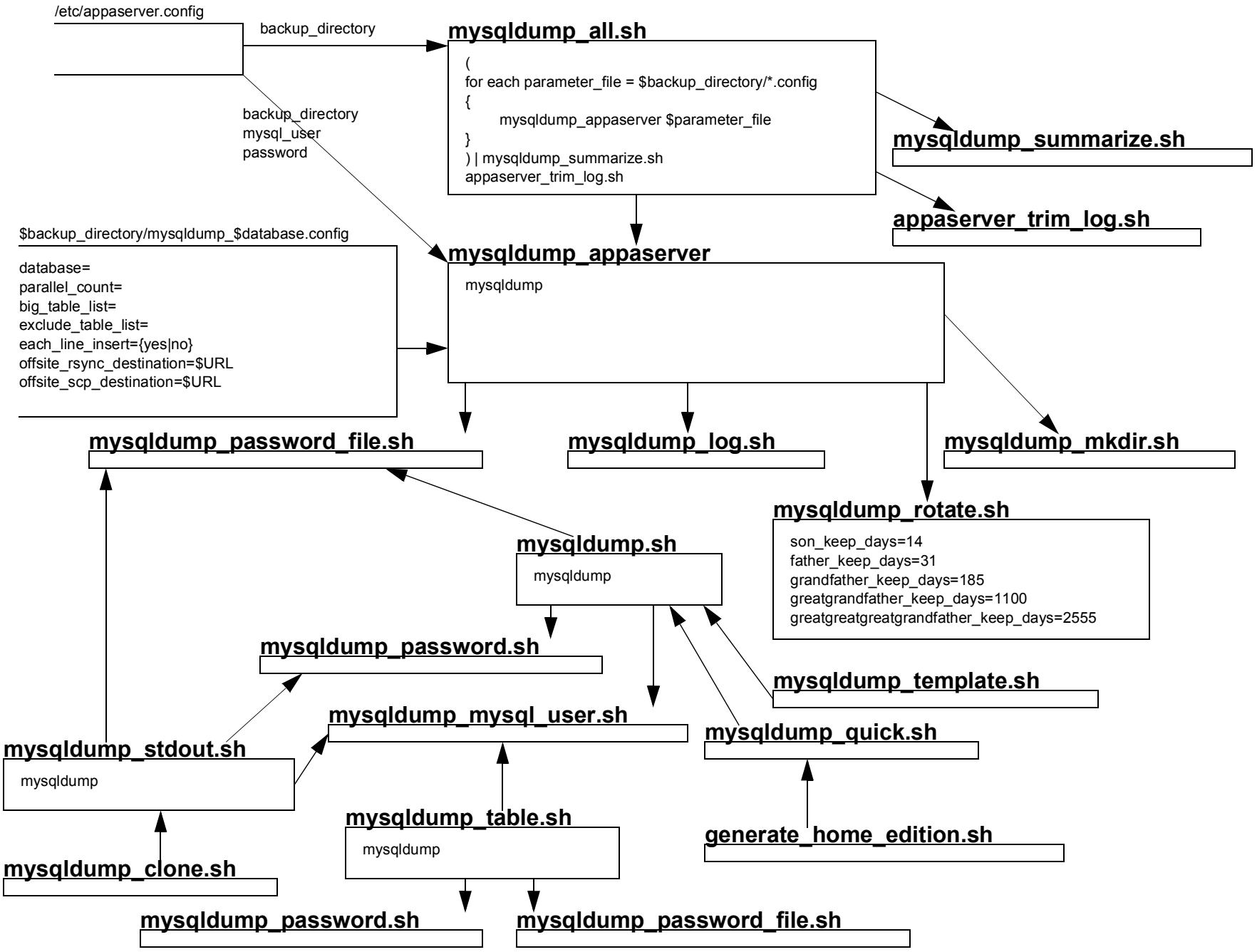
FORM_LOOKUP_DELETE

```
/* Usage */
FORM_LOOKUP_DELETE *form_lookup_delete_new(
    char *LOOKUP_DELETE_EXECUTE_LABEL,
    DICTIONARY *query_dictionary,
    char *lookup_delete_action_string() );

/* Process */
FORM_LOOKUP_DELETE *
    form_lookup_delete_calloc(
        void );

LIST *widget_container_list = list_new();

list_set(
    widget_container_list,
    ( WIDGET_CONTAINER *widget_container =
        widget_container_new(
        table_open, (char *)0 ) ) );

widget_container->table_open->border_boolean = 1;

list_set(
    widget_container_list,
    WIDGET_CONTAINER *widget_container_new(
        table_row, (char *)0 ) );

list_set(
    widget_container_list,
        WIDGET_CONTAINER *widget_container_new(
        table_data, (char *)0 ) );

WIDGET_CONTAINER *
    widget_confirm_checkbox_container(
        LOOKUP_DELETE_EXECUTE_LABEL,
        "Execute" /* checkbox_prompt */ );

list_set(
    widget_container_list,
    widget_confirm_checkbox_container() );

list_set(
    widget_container_list,
        WIDGET_CONTAINER *widget_container_new(
        table_data, (char *)0 ) );

list_set(
    widget_container_list,
    WIDGET_CONTAINER *widget_container_new(
        non_edit_text,
        FORM_LOOKUP_DELETE_OMIT_MESSAGE ) );

list_set(
    widget_container_list,
    WIDGET_CONTAINER *widget_container_new(
        table_row, (char *)0 ) );

list_set(
    widget_container_list,
        WIDGET_CONTAINER *widget_container_new(
        table_data, (char *)0 ) );

WIDGET_CONTAINER *
    widget_submit_button_container();

list_set(
    widget_container_list,
    widget_submit_button_container() );

list_set(
    widget_container_list,
        WIDGET_CONTAINER *widget_container_new(
        table_close, (char *)0 ) );

char *widget_container_list_html(
    (char *)0 /* state */,
    -1 /* row_number to omit index suffix */,
    (char *)0 /* background_color */,
    widget_container_list );

char *form_tag(
    FORM_LOOKUP_DELETE_NAME,
    lookup_delete_action_string,
    FRAMESET_TABLE_FRAME /* target_frame */ );

char *query_dictionary_hidden_html =
    dictionary_separate_hidden_html(
        DICTIONARY_SEPARATE_QUERY_PREFIX,
        query_dictionary );

char *form_html(
    form_tag(),
    widget_container_list_html(),
    query_dictionary_hidden_html,
    form_close_tag() );

/* Attributes */
LIST *widget_container_list;
WIDGET_CONTAINER *
    widget_confirm_checkbox_container;
WIDGET_CONTAINER *
    widget_submit_button_container;
char *widget_container_list_html;
char *form_tag;
char *query_dictionary_hidden_html;
char *form_html;
```



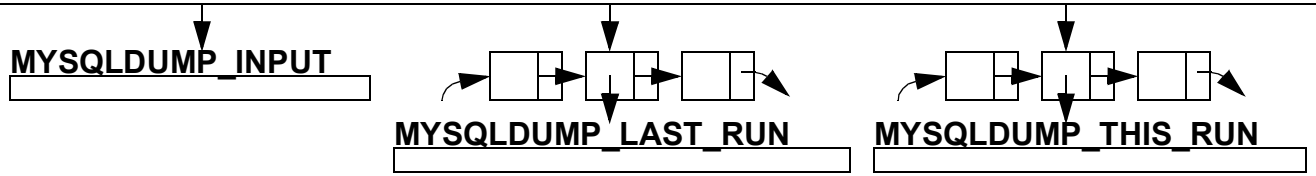


MYSQLDUMP Overview



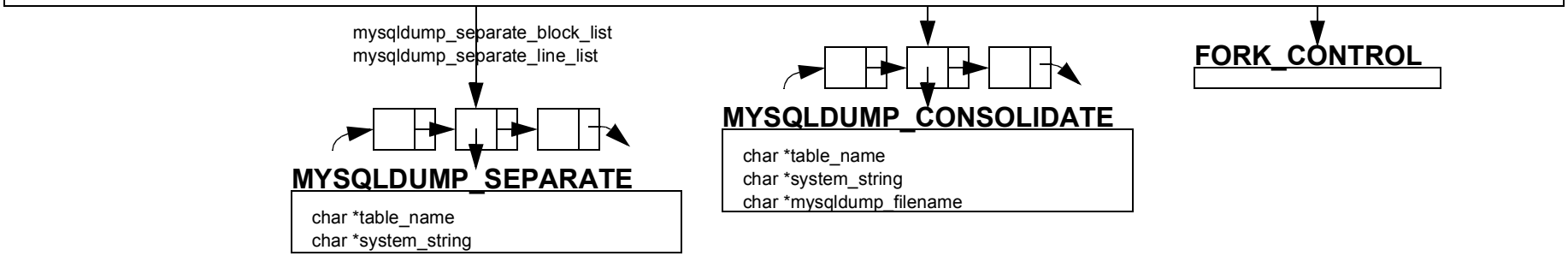
MYSQLDUMP (1)

```
/* Usage */
MYSQLDUMP *mysqldump_new(
    char *argv_0,
    char *config_filename );

/* Process */
MYSQLDUMP *mysqldump_calloc(
    void );

environ_umask( APPLICATION_UMASK );

MYSQLDUMP_INPUT *mysqldump_input =
    mysqldump_input_fetch(
        argv_0,
        config_filename );

boolean mysqldump_subdirectory_exists_boolean(
    char *mysqldump_input->son );

if ( !mysqldump_subdirectory_exists_boolean() )
{
    char *mysqldump_mkdir_system_string(
        const char *MYSQLDUMP_MKDIR_SCRIPT,
        char *mysqldump_input->database,
        char *mysqldump_input->backup_directory );

    system( mysqldump_mkdir_system_string() );
}
else
{
    char *mysqldump_last_run_filename(
        MYSQLDUMP_COUNT_PREFIX,
        mysqldump_input->date_yyyymmdd,
        mysqldump_input->son );

    if ( mysqldump_last_run_filename() )
    {
        char *mysqldump_last_run_yyyymmdd(
            mysqldump_last_run_filename() );

        LIST *mysqldump_last_run_list(
            MYSQLDUMP_DELIMITER,
            mysqldump_last_run_filename() );
    }
}

LIST *mysqldump_this_run_list(
    MYSQLDUMP_DELIMITER,
    mysqldump_input->database,
    mysqldump_last_run_list() );

char *mysqldump_this_run_threshold_display(
    MYSQLDUMP_DROP_THRESHOLD,
    msqldump_input->database,
    mysqldump_this_run_list() );

char *mysqldump_this_run_display_system_string(
    mysqldump_input->database,
    mysqldump_input->
        mysqldump_this_run_filename );

LIST *mysqldump_separate_block_list =
    LIST *mysqldump_separate_list(
        mysqldump_input->mysql_user,
        mysqldump_input->password_filename,
        mysqldump_input->date_yyyymmdd,
        mysqldump_input->separate_block_table_list,
        0 /* not is_line */ );

LIST *mysqldump_separate_line_list =
    LIST *mysqldump_separate_list(
        mysqldump_input->mysql_user,
        mysqldump_input->password_filename,
        mysqldump_input->date_yyyymmdd,
        mysqldump_input->separate_line_table_list,
        1 /* is_line */ );

LIST *mysqldump_consolidate_list(
    mysqldump_input->mysql_user,
    mysqldump_input->password_filename,
    mysqldump_input->date_yyyymmdd,
    mysqldump_input->separate_block_table_list,
    mysqldump_input->separate_line_table_list,
    mysqldump_input->exclude_table_list,
    mysqldump_this_run_list() );

FORK_CONTROL *mysqldump_fork_control(
    mysqldump_input->parallel_count,
    mysqldump_separate_block_list,
    mysqldump_separate_line_list,
    mysqldump_consolidate_list() );

char *mysqldump_remove_password_system_string(
    char *mysqldump_input->password_filename );

char *mysqldump_consolidate_tar_filename(
    mysqldump_input->database,
    mysq1dump_input->date_yyyymmdd );

char *mysqldump_consolidate_tar_system_string(
    mysqldump_consolidate_list(),
    mysqldump_consolidate_tar_filename() );

char *mysqldump_consolidate_rm_system_string(
    mysqldump_consolidate_list() );

char *mysqldump_log_system_string(
    const char *MYSQLDUMP_LOG_SCRIPT,
    const char *APPLICATION_LOG_EXTENSION,
    char *mysqldump_input->database,
    char *mysqldump_input->
        appaserver_error_directory,
    char *mysqldump_input->date_yyyymmdd,
    char *mysqldump_input->son );

char *mysqldump_rotate_system_string(
    const char MYSQLDUMP_ROTATE_SCRIPT,
    char *mysqldump_input->date_yyyymmdd,
    char *mysqldump_input->son,
    char *mysqldump_input->father,
    char *mysqldump_input->grandfather,
    char *mysqldump_input->greatgrandfather,
    char *mysqldump_input->greatgreatgrandfather );

char *mysqldump_rsync_system_string(
    mysqldump_input->son,
    mysqldump_input->offsite_rsync_destination );

char *mysqldump_scp_system_string(
    mysqldump_input->database,
    mysqldump_input->offsite_scp_destination,
    mysqldump_consolidate_tar_filename() );
```



MYSQLDUMP (2)

```
/* Usage */
char *mysqldump_filename(
    char *date_yyyymmdd,
    char *table_name,
    boolean is_gzip );

/* Process */
if ( is_gzip )
{
    sprintf(
        filename[],
        "%s_%s.sql.gz",
        table_name,
        date_yyyymmdd );
}
else
{
    sprintf(
        filename[],
        "%s_%s.sql",
        table_name,
        date_yyyymmdd );
}

/* Usage */
char *mysqldump_system_string(
    char *database,
    char *mysql_user,
    char *password_filename,
    char *table_name,
    boolean is_line );

/* Process */
char system_string[];
char *ptr = system_string;

ptr += sprintf( ptr,
    "/usr/bin/nice -n 9 "
    "mysqldump "
    "--defaults-extra-file=%s "
    "-u %s "
    "--force "
    "--quick "
    "--add-drop-table "
    "--extended-insert=",
    password_filename,
    mysql_user );

    if ( is_line )
        ptr += sprintf( ptr, "TRUE --complete-insert " );
    else
        ptr += sprintf( ptr, "FALSE " );

ptr += sprintf( ptr,
    "%s %s",
    database,
    table_name );

/* Usage */
FORK_CONTROL *mysqldump_fork_control(
    int mysqldump_input->parallel_count,
    LIST *mysqldump_separate_block_list,
    LIST *mysqldump_separate_line_list,
    LIST *mysqldump_consolidate_list );

/* Process */
FORK_CONTROL *fork_control =
    fork_control_new(
        parallel_count );

for MYSQLDUMP_SEPARATE *mysqldump_separate
    in mysqldump_separate_block_list
{
    void fork_control_process_set(
        fork_control->fork_control_process_list /* in/out */,
        mysqldump_separate->system_string
            /* command_line */ );
}

for MYSQLDUMP_SEPARATE *mysqldump_separate
    in mysqldump_separate_line_list
{
    void fork_control_process_set(
        fork_control->fork_control_process_list /* in/out */,
        mysqldump_separate->system_string
            /* command_line */ );
}

for MYSQLDUMP_CONSOLIDATE *
    mysqldump_consolidate
        in mysqldump_consolidate_list
{
    void fork_control_process_set(
        fork_control->fork_control_process_list /* in/out */,
        mysqldump_consolidate->system_string
            /* command_line */ );
}
```



```
/* Usage */
char *mysqldump_rsync_system_string(
    char *mysqldump_input->son,
    char *mysqldump_input->offsite_rsync_destination );

/* Process */
sprintf(
    system_string[],
    "rsync -aq --delete %s/ %s 1>&2",
    son,
    offsite_rsync_destination );

/* Usage */
char *mysqldump_scp_system_string(
    char *database,
    char *offsite_scp_destination,
    char *mysqldump_consolidate_tar_filename );

/* Process */
sprintf(
    system_string[],
    "rsync -aq %s %s%s.tar.gz",
    mysqldump_consolidate_tar_filename,
    offsite_scp_destination,
    database );

/* Driver */
int chdir( mysqldump->mysqldump_input->son );

if ( chdir() != 0 ) exit( 1 );

void fork_control_execute(
    mysqldump->fork_control->parallel_count,
    mysqldump->
        fork_control->
        fork_control_process_list );

system(
    mysqldump->
        remove_password_system_string );

if ( mysqldump->
        mysqldump_consolidate_tar_system_string )
{
    system(
        mysqldump->
            mysqldump_consolidate_tar_system_string );
}

if ( mysqldump->
        mysqldump_consolidate_rm_system_string )
{
    system(
        mysqldump->
            mysqldump_consolidate_rm_system_string );
}

system( mysqldump->log_system_string );

system( mysqldump->rotate_system_string );

if ( mysqldump->rsync_system_string )
{
    system( mysqldump->rsync_system_string );
}

if ( mysqldump->scp_system_string )
{
    system( mysqldump->scp_system_string );
}

if ( mysqldump->
        mysqldump_this_run_threshold_display )
{
    printf(
        "%s\n",
        mysqldump->
            mysqldump_this_run_threshold_display );
}

system(
    mysqldump->
        mysqldump_this_run_display_system_string );

/* Attributes */
MYSQLDUMP_INPUT *mysqldump_input;
boolean subdirectory_exists_boolean;
char *mkdir_system_string;
char *mysqldump_last_run_filename;
char *mysqldump_last_run_yyyymmdd;
LIST *mysqldump_last_run_list;
LIST *mysqldump_this_run_list;
char *mysqldump_this_run_threshold_display;
char *mysqldump_this_run_display_system_string;
LIST *mysqldump_separate_block_list;
LIST *mysqldump_separate_line_list;
LIST *mysqldump_consolidate_list;
FORK_CONTROL *fork_control;
char *remove_password_system_string;
char *mysqldump_consolidate_tar_filename;
char *mysqldump_consolidate_tar_system_string;
char *mysqldump_consolidate_rm_system_string;
char *log_system_string;
char *rotate_system_string;
char *rsync_system_string;
char *scp_system_string;
```

MYSQLDUMP_INPUT

```
/* Usage */
MYSQLDUMP_INPUT *mysqldump_input_fetch(
    char *argv_0,
    char *config_filename );

/* Process */
MYSQLDUMP_INPUT *mysqldump_input_calloc(
    void );

void mysqldump_input_parse(
    char *config_filename,
    char **database /* out */,
    int *parallel_count /* out */,
    char **separate_block_table_list_string /* out */,
    char **separate_line_table_list_string /* out */,
    char **exclude_table_list_string /* out */,
    char **offsite_rsync_destination /* out */,
    char **offsite_scp_destination  /* out */ );

void mysqldump_input_appaserver_parse(
    char *database,
    char **mysql_user /* out */,
    char **password /* out */,
    char **log_directory /* out */,
    char **backup_directory /* out */ );

char *mysqldump_input_password_filename(
    MYSQLDUMP_PASSWORD_FILE_SCRIPT,
    password );

char *mysqldump_input_date_yyyymmdd(
    void );

char *mysqldump_input_basename(
    char *argv_0 );

char *mysqldump_input_son(
    char *database,
    char *backup_directory );

char *mysqldump_input_father(
    char *database,
    char *backup_directory );

char *mysqldump_input_grandfather(
    char *database,
    char *backup_directory );

char *mysqldump_input_greatgrandfather(
    char *database,
    char *backup_directory );

char *mysqldump_input_greatgreatgrandfather(
    char *database,
    char *backup_directory );

LIST *mysqldump_input_separate_block_table_list(
    char *separate_block_table_list_string );

LIST *mysqldump_input_separate_line_table_list(
    char *separate_line_table_list_string );

LIST *mysqldump_input_exclude_table_list(
    char *exclude_table_list_string );

char *mysqldump_this_run_filename(
    MYSQLDUMP_COUNT_PREFIX,
    mysqldump_input_date_yyyymmdd(),
    mysqldump_input_son() );

/* Usage */
char *mysqldump_input_password_filename(
    const char *MYSQLDUMP_PASSWORD_FILE_SCRIPT,
    char *mysql_password );

/* Process */
char *password_filename =
    string_pipe_fetch(
        MYSQLDUMP_PASSWORD_FILE_SCRIPT );

FILE *appaserver_output_file(
    password_filename );

void fprintf(
    appaserver_output_file(),
    "[mysqldump]\npassword=%s\n",
    mysql_password );

void fclose( appaserver_output_file() );

/* Attributes */
char *database;
int parallel_count;
char *separate_block_table_list_string;
char *separate_line_table_list_string;
char *exclude_table_list_string;
char *offsite_rsync_destination;
char *offsite_scp_destination;
char *mysql_user;
char *password;
char *log_directory;
char *backup_directory;
char *password_filename;
char *date_yyyymmdd;
char *basename;
char *son;
char *father;
char *grandfather;
char *greatgrandfather;
char *greatgreatgrandfather;
LIST *separate_block_table_list;
LIST *separate_line_table_list;
LIST *exclude_table_list;
char *mysqldump_this_run_filename;
```

MYSQLDUMP_LAST_RUN

```
/* Usage */
LIST *mysqldump_last_run_list(
    char MYSQLDUMP_DELIMITER,
    char *mysqldump_last_run_filename() );

/* Process */
LIST *list = list_new();

FILE *appaserver_input_file(
    mysqldump_last_run_filename );

for char string_input( input[], appaserver_input_file ) )
{
    piece(
        table_name[],
        MYSQLDUMP_DELIMITER,
        input,
        0 );

    piece(
        row_count_string[],
        MYSQLDUMP_DELIMITER,
        input,
        1 );

    list_set(
        list,
        mysqldump_last_run_new(
            strdup( table_name ),
            string_atoi( row_count_string ) ) );
}

void fclose( appaserver_input_file() );

/* Usage */
MYSQLDUMP_LAST_RUN *mysqldump_last_run_new(
    char *table_name,
    int row_count );

/* Process */
MYSQLDUMP_LAST_RUN *
    mysqldump_last_run_calloc(
        void );

/* Usage */
int mysqldump_last_run_row_count(
    LIST *mysqldump_last_run_list(),
    char *table_name );

/* Process */
MYSQLDUMP_LAST_RUN *mysqldump_last_run_seek(
    LIST *mysqldump_last_run_list,
    char *table_name );

/* Usage */
char *mysqldump_last_run_filename(
    char *MYSQLDUMP_COUNT_PREFIX,
    char *mysqldump_input->date_yyyymmdd,
    char *mysqldump_input->son );

/* Process */
sprintf(
    filename,
    "ls -1 %s/%s_*.dat 2>/dev/null | grep -v %s | tail -1",
    son,
    MYSQLDUMP_COUNT_PREFIX,
    date_yyyymmdd );

/* Usage */
char *mysqldump_last_run_yyyymmdd(
    char *mysqldump_last_run_filename() );

/* Attributes */
char *table_name;
int row_count;
```

MYSQLDUMP_THIS_RUN (1)

```
/* Usage */
LIST *mysqldump_this_run_list(
    char MYSQLDUMP_DELIMITER,
    char *mysqldump_input->database,
    char *mysqldump_input->
        mysqldump_this_run_filename,
    LIST *mysqldump_last_run_list() );

/* Process */
LIST *list = list_new();

void environment_database_set( database );

char *mysqldump_this_run_system_string(
    mysqldump_this_run_filename );

FILE *appaserver_input_pipe(
    mysqldump_this_run_system_string() );

for char string_input( input[], appaserver_input_file ) )
{
    piece(
        table_name[],
        MYSQLDUMP_DELIMITER,
        input,
        0 );

    piece(
        row_count_string[],
        MYSQLDUMP_DELIMITER,
        input,
        1 );

    MYSQLDUMP_THIS_RUN *mysqldump_this_run =
        mysqldump_this_run_new(
            mysqldump_last_run_list,
            strdup( table_name ),
            string_atoi( row_count_string ) );

    list_set(
        list,
        mysqldump_this_run );
}

void pclose( appaserver_input_pipe() );

/* Usage */
MYSQLDUMP_THIS_RUN *mysqldump_this_run_new(
    LIST *mysqldump_last_run_list,
    char *table_name,
    int row_count );

/* Process */
MYSQLDUMP_THIS_RUN *
    mysqldump_this_run_calloc(
        void );

double mysqldump_this_run_percentage_drop(
    mysqldump_last_run_list,
    table_name,
    row_count );

/* Attributes */
char *table_name;
int row_count;
double percentage_drop;
```

MYSQLDUMP_THIS_RUN (2)

```
/* Usage */
double mysqldump_this_run_percentage_drop(
    LIST *mysqldump_last_run_list,
    char *table_name,
    int row_count );

/* Process */
int mysqldump_last_run_row_count(
    mysqldump_last_run_list,
    table_name );

if ( !mysqldump_last_run_row_count() ) return 0.0;

int row_count_increase =
    row_count -
    mysqldump_last_run_row_count();

if ( row_count_increase >= 0 ) return 0.0;

int row_count_drop = -row_count_increase;

double percentage_drop =
    (double)row_count_drop /
    (double)mysqldump_last_run_row_count();

/* Usage */
char *mysqldump_this_run_threshold_display(
    const double MYSQLDUMP_DROP_THRESHOLD,
    char *database,
    LIST *mysqldump_this_run_list() );

/* Process */
LIST *mysqldump_this_run_threshold_name_list(
    const double MYSQLDUMP_DROP_THRESHOLD,
    LIST *mysqldump_this_run_list );

if ( list_length(
        mysqldump_this_run_threshold_name_list() )
{
    snprintf(
        display[],
        sizeof ( display ),
        "%s warning: Row count dropped ... %d ...: %s",
        database,
        (int)MYSQLDUMP_DROP_THRESHOLD * 100.0 ),
        list_display_delimited(
            mysqldump_this_run_threshold_name_list(),
            ',' ) );
}

/* Usage */
char *mysqldump_this_run_filename(
    const char *MYSQLDUMP_COUNT_PREFIX,
    char *mysqldump_input_date_yyyymmdd(),
    char *mysqldump_input_son() );

/* Process */
snprintf(
    filename[],
    sizeof ( filename ),
    "%s/%s_%s.dat",
    mysqldump_input_son,
    MYSQLDUMP_COUNT_PREFIX,
    mysqldump_input_date_yyyymmdd );

/* Usage */
char *mysqldump_this_run_system_string(
    char *mysqldump_this_run_filename() );

/* Process */
snprintf(
    system_string[],
    sizeof ( system_string ),
    "table_count.e | "
    "sort_delimited_numbers.e '=' y | "
    "tee %s",
    mysqldump_this_run_filename );

/* Usage */
char *mysqldump_this_run_display_system_string(
    char *database,
    char *mysqldump_this_run_filename );

/* Process */
snprintf(
    system_string[],
    sizeof ( system_string ),
    "/bin/echo -e \"\\nDatabase: %s\"; cat %s",
    database,
    mysqldump_this_run_filename );
```

MYSQLDUMP_SEPARATE

```
/* Usage */
LIST *mysqldump_separate_list(
    char *mysqldump_input->database,
    char *mysqldump_input->mysql_user,
    char *mysqldump_input->password_filename,
    char *mysqldump_input->date_yyyymmdd,
    LIST *mysqldump_input->separate_table_list,
    boolean is_line );

/* Process */
LIST *list = list_new();

for char *table_name in separate_table_list
{
    list_set(
        list,
        mysqldump_separate_new(
            database,
            mysql_user,
            password_filename,
            date_yyyymmdd,
            table_name,
            is_line ) );
}
```

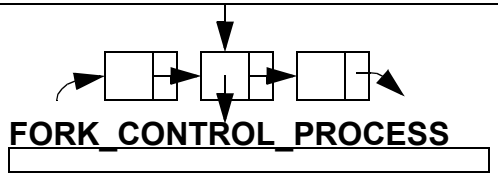
```
/* Usage */
MYSQLDUMP_SEPARATE *
    mysqldump_separate_new(
        char *database,
        char *mysql_user,
        char *password_filename,
        char *date_yyyymmdd,
        char *table_name,
        boolean is_line );

/* Process */
MYSQLDUMP_SEPARATE *
    mysqldump_separate_calloc(
        void );

char *mysqldump_system_string(
    database,
    mysql_user,
    password_filename,
    table_name,
    is_line );

char *mysqldump_filename(
    date_yyyymmdd,
    table_name,
    1 /* is_gzip */ );

char *mysqldump_separate_system_string(
    mysqldump_system_string(),
    mysqldump_filename() );

/* Usage */
char *mysqldump_separate_system_string(
    char *mysqldump_system_string(),
    char *mysqldump_filename() );

/* Process */
void sprintf(
    system_string[],
    “%s | gzip > %s”,
    mysqldump_system_string,
    mysqldump_filename );

/* Attributes */
char *table_name;
char *mysqldump_filename;
char *mysqldump_system_string;
char *system_string;
```

MYSQLDUMP_CONSOLIDATE

```
/* Usage */
LIST *mysqldump_consolidate_list(
    char *mysqldump_input->database,
    char *mysqldump_input->mysql_user,
    char *mysqldump_input->password_filename,
    char *mysqldump_input->date_yyyymmdd,
    LIST *mysqldump_input->separate_block_table_list,
    LIST *mysqldump_input->separate_line_table_list,
    LIST *mysqldump_input->exclude_table_list,
    LIST *mysqldump_this_run_list() );

/* Process */
LIST *list = list_new();

for MYSQLDUMP_THIS_RUN *mysqldump_this_run
    in mysqldump_this_run_list
{
    if ( list_exists_string(
            mysqldump_this_run->table_name,
            separate_block_table_list ) )
    {
        continue;
    }

    if ( list_exists_string(
            mysqldump_this_run->table_name,
            separate_line_table_list ) )
    {
        continue;
    }

    if ( list_exists_string(
            mysqldump_this_run->table_name,
            exclude_table_list ) )
    {
        continue;
    }

    list_set(
        list,
        mysqldump_consolidate_new(
            database,
            mysql_user,
            password_filename,
            date_yyyymmdd,
            mysqldump_this_run->table_name ) );
}

/* Usage */
MYSQLDUMP_CONSOLIDATE *
    mysqldump_consolidate_new(
        char *database,
        char *mysql_user,
        char *password_filename,
        char *date_yyyymmdd,
        char *table_name );

/* Process */
MYSQLDUMP_CONSOLIDATE *
    mysqldump_consolidate_calloc(
        void );

char *mysqldump_system_string(
    database,
    mysql_user,
    password_filename,
    table_name,
    1 /* is_line */ );

char *mysqldump_filename(
    date_yyyymmdd,
    table_name,
    0 /* not is_gzip */ );

char *mysqldump_consolidate_system_string(
    mysqldump_system_string(),
    mysqldump_filename() );

/* Usage */
char *mysqldump_consolidate_system_string(
    char *mysqldump_system_string(),
    char *mysqldump_filename() );

/* Process */
void sprintf(
    system_string[],
    "%s > %s",
    mysqldump_system_string,
    mysqldump_filename );

/* Usage */
char *mysqldump_consolidate_tar_filename(
    char *database,
    char *date_yyyymmdd );

/* Process */
void sprintf( filename[],
    "%s_%s.tar.gz",
    database,
    date_yyyymmdd );

/* Usage */
char *mysqldump_consolidate_tar_system_string(
    LIST *mysqldump_consolidate_list(),
    char *mysqldump_consolidate_tar_filename() );

/* Process */
char system_string[];
char *ptr = system_string;

ptr += sprintf( ptr,
    "tar -czf %s",
    mysqldump_consolidate_tar_filename );

for MYSQLDUMP_CONSOLIDATE *
    mysqldump_consolidate
        in mysqldump_consolidate_list
{
    ptr += sprintf( ptr,
        " %s",
        mysqldump_consolidate->mysql_filename );
}

/* Usage */
char *mysqldump_consolidate_rm_system_string(
    LIST *mysqldump_consolidate_list() );

/* Process */
char system_string[];
char *ptr = system_string;

ptr += sprintf( ptr, "rm" );

for MYSQLDUMP_CONSOLIDATE *
    mysqldump_consolidate
        in mysqldump_consolidate_list
{
    ptr += sprintf( ptr,
        " %s",
        mysqldump_consolidate->mysql_filename );
}

/* Attributes */
char *table_name;
char *mysqldump_filename;
char *mysqldump_system_string;
char *system_string;
```

FORK_CONTROL

```
/* Usage */
FORK_CONTROL *fork_control_new(
    int parallel_count );

/* Process */
FORK_CONTROL *fork_control_calloc( void );

/* Driver */
void fork_control_execute(
    int fork_control->parallel_count,
    LIST *fork_control->fork_control_process_list );

/* Process */
int process_count = 0;

while ( 1 )
{
    if ( ( FORK_CONTROL_PROCESS *
                fork_control_process =
                    fork_control_process_next(
                        fork_control_process_list ) ) )
    {
        fork_control_process_execute(
            &fork_control_process->
                fork_control_process_state,
            &fork_control_process->
                running_process_id,
            fork_control_process->command_line );

        process_count++
    }

    if ( !process_count ) break;

    if ( process_count < parallel_count )
    {
        if ( fork_control_process_pending_boolean(
                fork_control_process_list ) )
        {
            continue;
        }
    }

    if ( ( pid_t done_process_id = wait( (int *)0 ) ) )
    {
        FORK_CONTROL_PROCESS *
            fork_control_process =
                fork_control_process_seek(
                    fork_control_process_list,
                    done_process_id );

        fork_control_process->
            fork_control_process_state =
                fork_control_process_state_done;

        process_count--;
    }
}

/* Public */
void fork_control_process_list_display(
    LIST *fork_control_process_list );

/* Attributes */
int parallel_count;
LIST *fork_control_process_list;
```



FORK_CONTROL_PROCESS

FORK_CONTROL_PROCESS

```
/* Usage */
FORK_CONTROL_PROCESS *fork_control_process_new(
    char *command_line );

/* Process */
FORK_CONTROL_PROCESS *
    fork_control_process_calloc( void );

fork_control_process->
    fork_control_process_state =
        fork_control_process_state_initial;

/* Usage */
void fork_control_process_set(
    LIST *fork_control_process_list /* in/out */,
    char *command_line );

/* Process */
list_set(
    fork_control_process_list,
    fork_control_process_new(
            commnd_line ) );

/* Usage */
FORK_CONTROL_PROCESS *fork_control_process_next(
    LIST *fork_control_process_list );

/* Process */
for FORK_CONTROL_PROCESS *
    fork_control_process
        int fork_control_process_list
{
    if ( fork_control_process->
            fork_control_process_state ==
                fork_control_process_initial )
    {
        return fork_control_process;
    }
}
return NULL;

/* Usage */
void fork_control_process_execute(
    enum fork_control_process_state *
        fork_control_process_state /* out */,
    pid_t *running_process_id /* out */,
    char *command_line );

/* Process */
if ( ( *running_process_id = fork() ) == 0 )
{
    execlp(
        "/bin/sh",
        "sh",
        "-c",
        command_line,
        (char *)0 );
}
else
{
    *fork_control_process_state =
        fork_control_process_executing;
}

/* Usage */
boolean fork_control_process_pending_boolean(
    LIST *fork_control_process_list );

/* Process */
for FORK_CONTROL_PROCESS *
    fork_control_process
        in fork_control_process_list
{
    if ( fork_control_process->
            fork_control_process_state ==
                fork_control_process_initial )
    {
        return 1;
    }
}
return 0;

/* Usage */
FORK_CONTROL_PROCESS *
    fork_control_process_seek(
        LIST *fork_control_process_list,
        pid_t done_process_id );

/* Process */
for FORK_CONTROL_PROCESS *
    fork_control_process
        in fork_control_process_list
{
    if ( fork_control_process->
            running_process_id ==
                done_process_id )
    {
        return fork_control_process;
    }
}
return NULL;

/* Attributes */
char *command_line;
enum fork_control_process_state
    fork_control_process_state;
pid_t running_process_id;
```

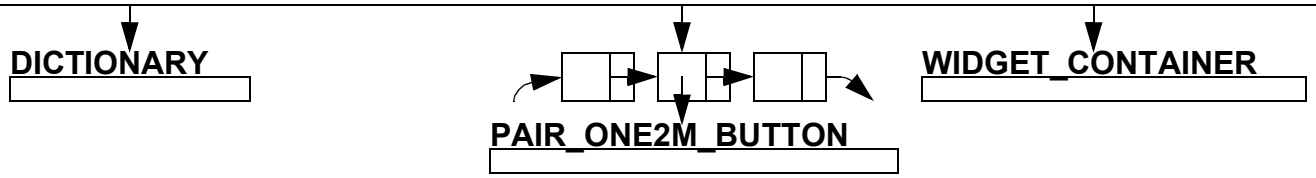
fork_control_process_initial
fork_control_process_executing
fork_control_process_done

STRING

```
/* Usage */
void string_replace_command_line(
    char *command_line,
    char *string,
    char *placeholder );

/* Process */
char *double_quotes_around(
    char destination[],
    string );

char *string_with_space_search_replace(
    command_line /* source_destination */,
    placeholder /* search */,
    double_quotes_around() /* replace */ );

/* Usage */
char *string_with_space_search_replace(
     char *source_destination,
     char *search_string,
     char *replace_string );

/* Process */

/* Public */
boolean string_strcpy(
     char *destination,
     char *source,
     int buffer_size );

boolean string_strcmp(
     char *string1,
     char *string2 );

char *string_escape_character(
     char *destination,
     char *data,
     char character );

char *string_escape_character_array(
     char *destination,
     char *data,
     char *character_array );

/* Position is 1 based */
int string_character_position(
     char *data,
     char character );

char *string_trim_character(
     char *data,
     char character );

char *string_trim_character_array(
     char *data,
     char *character_array );

char *string_decode_html_post(
     char *destination,
     char *source );

char *string_trim( char *string );

char *string_extract_lt_gt_delimited(
     char *destination,
     char *source );

int string_index( char *source );

char string_last_character( char *string );

char *string_up( char *string );

char *string_low( char *string );

char string_delimiter( char *string );

char *string_file_fetch(
     char *filename,
     char delimiter );

boolean string_file_write(
     char *filename,
     char *string );
```

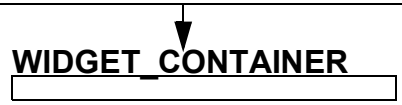
PAIR_ONE2M_PROMPT_INSERT

```
/* Usage */
PAIR_ONE2M_PROMPT_INSERT *
   pair_one2m_prompt_insert_new(
      const char *DICTIONARY_SEPARATE_PAIR_PREFIX,
      const char *PAIR_ONE2M_ONE_FOLDER_KEY,
      const char *PAIR_ONE2M_MANY_FOLDER_KEY,
      const char *PAIR_ONE2M_UNFULFILLED_LIST_KEY.
      const char PAIR_ONE2M_FOLDER_DELIMITER,
      char *one_folder_name,
      char *recall_save_javascript(),
      LIST *relation_one2m_pair_list() );

/* Process */
PAIR_ONE2M_PROMPT_INSERT *
   pair_one2m_prompt_insert_calloc(
      void );

DICTIONARY *pair_dictionary =
   dictionary_small();

void dictionary_set(
   pair_dictionary,
   PAIR_ONE2M_ONE_FOLDER_KEY,
   one_folder_name );

char *pair_one2m_prompt_insert_unfulfilled_list_string(
   const char PAIR_ONE2M_FOLDER_DELIMITER,
   LIST *relation_one2m_pair_list );

if ( pair_one2m_prompt_insert_unfulfilled_list_string() )
{
   void dictionary_set(
      pair_dictionary,
      PAIR_ONE2M_UNFULFILLED_LIST_KEY,
      pair_one2m_prompt_insert_unfulfilled_list_string();
}

LIST *pair_one2m_button_list(
   DICTIONARY_SEPARATE_PAIR_PREFIX,
   PAIR_ONE2M_MANY_FOLDER_KEY,
   recall_save_javascript,
   relation_one2m_pair_list );

WIDGET_CONTAINER *
   pair_one2m_prompt_insert_hidden_widget(
      char *DICTIONARY_SEPARATE_PAIR_PREFIX,
      char *PAIR_ONE2M_MANY_FOLDER_KEY );

/* Attributes */
DICTIONARY *pair_dictionary;
char *unfulfilled_list_string;
LIST *pair_one2m_button_list;
WIDGET_CONTAINER *hidden_widget;
```



PAIR_ONE2M_BUTTON

```
/* Usage */
LIST *pair_one2m_button_list(
    const char *DICTIONARY_SEPARATE_PAIR_PREFIX,
    const char *PAIR_ONE2M_MANY_FOLDER_KEY,
    char *recall_save_javascript(),
    LIST *relation_one2m_pair_list() );

/* Process */
LIST *list = list_new();

for RELATION_ONE2M *relation_one2m in
    relation_one2m_pair_list
{
    list_set(
        list,
        pair_one2m_button_new(
            DICTIONARY_SEPARATE_PAIR_PREFIX,
            PAIR_ONE2M_MANY_FOLDER_KEY,
            recall_save_javascript,
            relation_one2m->many_folder_name ) );
}

/* Usage */
PAIR_ONE2M_BUTTON *pair_one2m_button_new(
    const char *DICTIONARY_SEPARATE_PAIR_PREFIX,
    const char *PAIR_ONE2M_MANY_FOLDER_KEY,
    char *recall_save_javascript(),
    char *many_folder_name );

/* Process */
PAIR_ONE2M_BUTTON *pair_one2m_button_calloc(
    void );

WIDGET_CONTAINER *
    widget_container =
        widget_container_new(
            button,
            many_folder_name );

free( widget_container->widget_button );

widget_container->widget_button =
    widget_button_new(
        many_folder_name /* label */ );

char *pair_one2m_button_action_string(
    const char *DICTIONARY_SEPARATE_PAIR_PREFIX,
    const char *PAIR_ONE2M_MANY_FOLDER_KEY,
    char *recall_save_javascript,
    char *many_folder_name );

widget_container->
    widget_button->
    button->
    action_string =
        pair_one2m_button_action_string();

/* Attributes */
WIDGET_CONTAINER *widget_container;
char *action_string;
```



PAIR_ONE2M_POST_PROMPT_INSERT

```
/* Usage */
PAIR_ONE2M_POST_PROMPT_INSERT *
    pair_one2m_post_prompt_insert_new(
        const char *PAIR_ONE2M_ONE_FOLDER_KEY,
        const char *PAIR_ONE2M_MANY_FOLDER_KEY,
        const char *PAIR_ONE2M_UNFULFILLED_LIST_KEY.
        const char PAIR_ONE2M_FOLDER_DELIMITER,
        DICTIONARY *dictionary_separate->
            pair_dictionary );

/* Process */
PAIR_ONE2M_POST_PROMPT_INSERT *
    pair_one2m_post_prompt_insert_calloc(
        void );

char *one_folder_name =
    dictionary_get(
        PAIR_ONE2M_ONE_FOLDER_KEY,
        pair_dictionary );

if ( !one_folder_name ) return this;

char *many_folder_name =
    dictionary_get(
        PAIR_ONE2M_MANY_FOLDER_KEY,
        pair_dictionary );

if ( !many_folder_name ) return this;

char *unfulfilled_list_string =
    dictionary_get(
        PAIR_ONE2M_UNFULFILLED_LIST_KEY,
        pair_dictionary );

LIST *unfulfilled_list =
    list_delimiter_extract(
        unfulfilled_list_string,
        PAIR_ONE2M_FOLDER_DELIMITER );

char *
    pair_one2m_post_prompt_insert_next_folder_name(
        char *many_folder_name,
        LIST *unfulfilled_list );

/* Attribute */
char *one_folder_name;
char *many_folder_name;
char *unfulfilled_list_string;
LIST *unfulfilled_list;
char *next_folder_name;
```

PAIR_ONE2M_TABLE_INSERT

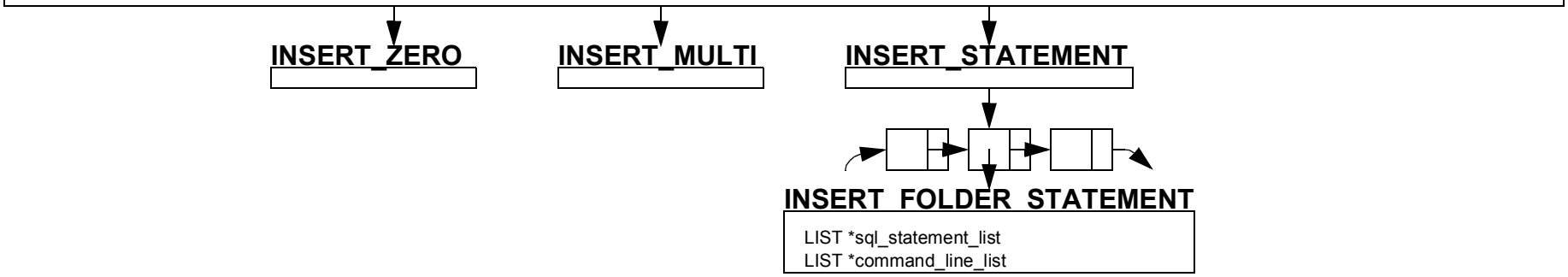
```
/* Usage */
PAIR_ONE2M_TABLE_INSERT *
    pair_one2m_table_insert_new(
        const char *PAIR_ONE2M_ONE_FOLDER_KEY,
        DICTIONARY *dictionary_separate->
            pair_dictionary );

/* Process */
PAIR_ONE2M_TABLE_INSERT *
    pair_one2m_table_insert_calloc(
        void );

char *one_folder_name =
    dictionary_get(
        PAIR_ONE2M_ONE_FOLDER_KEY,
        pair_dictionary );

if ( !one_folder_name ) return NULL;

LIST *folder_attribute_primary_key_list =
    folder_attribute_fetch_primary_key_list(
        one_folder_name );

/* Attribute */
char *one_folder_name;
LIST *folder_attribute_primary_key_list;
```

PAIR_ONE2M_POST_TABLE_INSERT

```
/* Usage */
PAIR_ONE2M_POST_TABLE_INSERT *
   pair_one2m_post_table_insert_new(
      DICTIONARY *dictionary_separate->
         pair_dictionary /* in/out */,
      const char *PAIR_ONE2M_UNFULFILLED_LIST_KEY,
      const char *PAIR_ONE2M_FULFILLED_LIST_KEY,
      const char PAIR_ONE2M_FOLDER_DELIMITER,
      char *many_folder_name );

/* Process */
PAIR_ONE2M_POST_TABLE_INSERT *
   pair_one2m_post_table_insert_calloc(
      void );

LIST *pair_one2m_post_table_insert_fulfilled_list(
   pair_dictionary,
   PAIR_ONE2M_FULFILLED_LIST_KEY,
   PAIR_ONE2M_FOLDER_DELIMITER );

boolean list_string_exists(
   many_folder_name,
   pair_one2m_post_table_insert_fulfilled_list() );

if ( !list_string_exists() )
{
   void list_set(
      pair_one2m_post_table_insert_fulfilled_list(),
      many_folder_name );

   char *fulfilled_list_string =
      list_delimited_string(
         pair_one2m_post_table_insert_fulfilled_list(),
         PAIR_ONE2M_FOLDER_DELIMITER );

   void dictionary_set(
      pair_dictionary,
      PAIR_ONE2M_FULFILLED_LIST_KEY,
      fulfilled_list_string );
}

LIST *pair_one2m_post_table_insert_unfulfilled_list(
   pair_dictionary,
   PAIR_ONE2M_UNFULFILLED_LIST_KEY,
   PAIR_ONE2M_FOLDER_DELIMITER );

list_subtract_string(
   pair_one2m_post_table_insert_unfulfilled_list(),
   many_folder_name );

char *unfulfilled_list_string =
   list_delimited_string(
      pair_one2m_post_table_insert_unfulfilled_list(),
      PAIR_ONE2M_FOLDER_DELIMITER );

void dictionary_set(
   pair_dictionary,
   PAIR_ONE2M_UNFULFILLED_LIST_KEY,
   unfulfilled_list_string );

char *pair_one2m_post_table_insert_next_folder_name(
   pair_one2m_post_table_insert_unfulfilled_list() );

/* Usage */
LIST *pair_one2m_post_table_insert_fulfilled_list(
   DICTIONARY *pair_dictionary,
   const char *PAIR_ONE2M_FULFILLED_LIST_KEY,
   const char PAIR_ONE2M_FOLDER_DELIMITER );

/* Process */
char *fulfilled_list_string =
   dictionary_get(
      PAIR_ONE2M_FULFILLED_LIST_KEY,
      pair_dictionary );

if ( !fulfilled_list_string )
{
   fulfilled_list = list_new();
}
else
{
   LIST *fulfilled_list =
      list_string_list(
         fulfilled_list_string,
         PAIR_ONE2M_FOLDER_DELIMITER );
}

/* Usage */
LIST *pair_one2m_post_table_insert_unfulfilled_list(
   DICTIONARY *pair_dictionary,
   const char *PAIR_ONE2M_UNFULFILLED_LIST_KEY,
   const char PAIR_ONE2M_FOLDER_DELIMITER );

/* Process */
char *unfulfilled_list_string =
   dictionary_get(
      PAIR_ONE2M_UNFULFILLED_LIST_KEY,
      pair_dictionary );

LIST *unfulfilled_list =
   list_string_list(
      unfulfilled_list_string,
      PAIR_ONE2M_FOLDER_DELIMITER );

/* Usage */
char *pair_one2m_post_table_insert_next_folder_name(
   LIST *pair_one2m_post_table_insert_unfulfilled_list );

/* Process */
list_first(
   pair_one2m_post_table_insert_unfulfilled_list );
}

/* Attributes */
LIST *fulfilled_list;
LIST *unfulfilled_list;
char *fulfilled_list_string;
char *unfulfilled_list_string;
char *next_folder_name;
```

INSERT (1)

```
/* Usage */
INSERT *insert_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    LIST *root_primary_key_list,
    LIST *folder_attribute_append_isa_list,
    LIST *relation_mto1_list,
    LIST *relation_mto1_isa_list,
    DICTIONARY *prompt_dictionary /* in/out */,
    DICTIONARY *multi_row_dictionary /* in/out */,
    LIST *ignore_name_list,
    PROCESS *post_change_process );

/* Process */
INSERT *insert_calloc(
    void );

char *appaserver_error_filespecification(
    application_name );

LIST *folder_attribute_insert_required_name_list(
    folder_attribute_append_isa_list );

if ( !dictionary_length( multi_row_dictionary ) )
{
    INSERT_ZERO *insert_zero =
        insert_zero_new(
            folder_name,
            root_primary_key_list,
            folder_attribute_append_isa_list,
            relation_mto1_list,
            relation_mto1_isa_list,
            prompt_dictionary /* in/out */,
            ignore_name_list,
            post_change_process,
            folder_attribute_insert_required_name_list() );
}
else
{
    INSERT_MULTI *insert_multi =
        insert_multi_new(
            application_name,
            session_key,
            login_name,
            role_name,
            folder_name,
            root_primary_key_list,
            folder_attribute_append_isa_list,
            relation_mto1_list,
            relation_mto1_isa_list,
            prompt_dictionary /* in/out */,
            multi_row_dictionary /* in/out */,
            ignore_name_list,
            post_change_process,
            folder_attribute_insert_required_name_list() );
}

if ( !insert_zero && !insert_multi ) return NULL;

INSERT_STATEMENT *insert_statement =
    insert_statement_new(
        session_key,
        login_name,
        role_name,
        appaserver_error_filespecification(),
        insert_zero,
        insert_multi );

int insert_folder_statement_sql_list_length(
    insert_statement->insert_folder_statement_list );

char *insert_result_string(
    folder_name,
    relation_mto1_isa_list,
    insert_folder_statement_sql_list_length() );

/* Attributes */
char *appaserver_error_filespecification;
LIST *folder_attribute_insert_required_name_list;
INSERT_ZERO *insert_zero;
INSERT_MULTI *insert_multi;
INSERT_STATEMENT *insert_statement;
int insert_folder_statement_sql_list_length;
char *result_string;
```



INSERT (2)

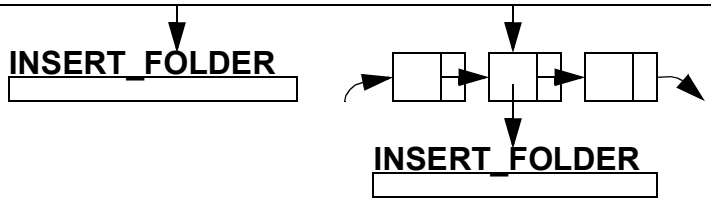
```
/* Usage */
INSERT *insert_automatic_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    LIST *folder->folder_attribute_primary_key_list,
    LIST *folder_attribute_append_isa_list(),
    LIST *relation_mto1_isa_list(),
    DICTIONARY *prompt_dictionary,
    DICTIONARY *multi_row_dictionary,
    PROCESS *post_change_process );

/* Process */
LIST *folder_attribute_non_primary_name_list(
    folder_attribute_append_isa_list );

DICTIONARY *insert_automatic_multi_row_dictionary(
    folder_attribute_non_primary_name_list(),
    multi_row_dictionary );

if ( dictionary_length(
        insert_automatic_multi_row_dictionary() ) )
{
    INSERT *insert_new(
        application_name,
        session_key,
        login_name,
        role_name,
        folder_name,
        folder_attribute_primary_key_list
            /* root_primary_key_list */,
        folder_attribute_append_isa_list,
        relation_mto1_isa_list,
        prompt_dictionary
            /* in/out */,
        insert_automatic_multi_row_dictionary()
            /* in/out */,
        (LIST *)0 /* ignore_name_list */,
        post_change_process );
}

/* Usage */
DICTIONARY *insert_automatic_multi_row_dictionary(
    LIST *folder_attribute_non_primary_name_list,
    DICTIONARY *multi_row_dictionary );

/* Process */
DICTIONARY *dictionary = dictionary_medium();

LIST *dictionary_key_list( multi_row_dictionary );

int dictionary_highest_index( multi_row_dictionary );

for int index = 1 to dictionary_highest_index()
{
    LIST *dictionary_index_populated_name_list(
        folder_attribute_non_primary_name_list,
        multi_row_dictionary,
        index );

    int list_length(
        dictionary_index_populated_name_list() );

    if ( !list_lenght() ) continue;

    void dictionary_index_set(
        dictionary /* out */,
        multi_row_dictionary,
        dictionary_key_list(),
        index );
}
```

INSERT (3)

```
/* Usage */
char *insert_result_string(
    char *folder_name,
    LIST *relation_mto1_isa_list(),
    int insert_folder_statement_sql_list_length() );

/* Process */
if ( list_length( relation_mto1_isa_list ) )
{
    LIST *relation_mto1_folder_name_list(
        relation_mto1_isa_list );
}

char *insert_result_folder_display(
    char *folder_name,
    LIST *relation_mto1_folder_name_list() );

/* Usage */
APPASERVER_USER *insert_appaserver_user(
    const char *
        APPASERVER_USER_FULL_NAME_COLUMN,
    char *login_name,
    LIST *role_attribute_exclude_insert_name_list(),
    LIST *folder_attribute_append_isa_list() );

/* Process */
if ( list_string_exists(
        APPASERVER_USER_FULL_NAME_COLUMN,
        role_attribute_exclude_insert_name_list ) )
{
    APPASERVER_USER *appaserver_user_login_fetch(
        login_name,
        0 /* not fetch_role_name_list */ );
}
else
{
    FOLDER_ATTRIBUTE *folder_attribute =
        folder_attribute_seek(
            (char *)0 /* folder_name */,
            APPASERVER_USER_FULL_NAME_COLUMN,
            folder_attribute_append_isa_list );

    if (    folder_attribute
    &&  folder_attribute->omit_insert )
    {
        APPASERVER_USER *appaserver_user_login_fetch(
            login_name,
            0 /* not fetch_role_name_list */ );
    }
}

/* Driver */
char *insert_statement_error_string =
    char *insert_statement_execute(
        insert->
            insert_statement->
            insert_folder_statement_list,
        insert->
            appaserver_error_filename );
```

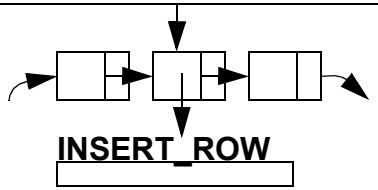
INSERT_ZERO

```
/* Usage */
INSERT_ZERO *insert_zero_new(
    char *folder_name,
    LIST *folder_attribute_primary_key_list,
    LIST *folder_attribute_append_isa_list,
    LIST *relation_mto1_list,
    LIST *relation_mto1_isa_list,
    DICTIONARY *prompt_dictionary /* in/out */,
    LIST *ignore_name_list,
    PROCESS *post_change_process,
    LIST *folder_attribute_insert_required_name_list );

/* Process */
INSERT_ZERO *insert_zero_calloc(
    void );

(void)relation_copy_new(
    prompt_dictionary /* in/out */,
    folder_attribute_append_isa_list,
    relation_mto1_list,
    0 /* row_number */ );

LIST *folder_attribute_name_list(
    folder_name,
    folder_attribute_append_isa_list );

INSERT_FOLDER *insert_folder =
    insert_folder_new(
        folder_name,
        folder_attribute_primary_key_list
            /* root_primary_key_list */,
        (LIST *)0 /* isa_primary_key_list */,
        folder_attribute_append_isa_list,
        prompt_dictionary,
        (DICTIONARY *)0 /* multi_row_dictionary */,
        ignore_name_list,
        post_change_process,
        folder_attribute_insert_required_name_list,
        0 /* row_number */,
        folder_attribute_name_list() );

if ( !insert_folder ) return NULL;

if ( !list_rewind( relation_mto1_isa_list ) )
        return this;

char *appaserver_parameter_mount_point();

LIST *insert_folder_isa_list = list_new();

for RELATION_MTO1 *
    relation_mto1 in
        relation_mto1_isa_list
{
    LIST *folder_attribute_name_list(
        relation_mto1->one_folder_name,
        folder_attribute_append_isa_list );

    if ( relation_mto1->
            one_folder->
            post_change_process_name )
    {
            PROCESS *post_change_process =
                process_fetch(
                    relation_mto1->
                        one_folder->
                        post_change_process_name,
                    (char *)0 /* document_root */,
                    (char *)0 /* relative_source_directory */,
                    1 /* check_executable_inside */,
                    appaserver_parameter_mount_point() );
    }
    else
    {
        post_change_process = NULL;
    }

    list_set(
        insert_folder_isa_list,
        insert_folder_new(
            relation_mto1->one_folder_name,
            folder_attribute_primary_key_list
                /* root_primary_key_list */,
            relation_mto1->
                one_folder->
                folder_attribute_primary_key_list
                /* isa_primary_key_list */,
            folder_attribute_append_isa_list,
            prompt_dictionary,
            (DICTIONARY *)0 /* multi_row_dictionary */,
            ignore_name_list,
            post_change_process,
            folder_attribute_insert_required_name_list,
            0 /* row_number */,
            folder_attribute_name_list() ) );
}

/* Attributes */
INSERT_FOLDER *insert_folder;
LIST *insert_folder_isa_list;
```



INSERT_MULTI (1)

```
/* Usage */
INSERT_MULTI *insert_multi_new(
    char *folder_name,
    LIST *root_primary_key_list,
    LIST *folder_attribute_append_isa_list,
    LIST *relation_mto1_list,
    LIST *relation_mto1_isa_list,
    DICTIONARY *prompt_dictionary /* in/out */,
    DICTIONARY *multi_row_dictionary /* in/out */,
    LIST *ignore_name_list,
    PROCESS *post_change_process,
    LIST *folder_attribute_insert_required_name_list );

/* Process */
INSERT_MULTI *insert_multi_calloc(
    void );

int dictionary_highest_index( multi_row_dictionary );

if ( dictionary_highest_index() < 1 ) return NULL;

(void)relation_copy_new(
    prompt_dictionary /* in/out */,
    folder_attribute_append_isa_list,
    relation_mto1_list,
    0 /* row_number */ );

for (  row_number = 1;
       row_number <= dictionary_highest_index();
       row_number++ )
{
    void insert_multi_attribute_default_set(
        folder_attribute_append_isa_list,
        multi_row_dictionary /* in/out */,
        row_number );

    (void)relation_copy_new(
        multi_row_dictionary /* in/out */,
        folder_attribute_append_isa_list,
        relation_mto1_list,
        row_number );

    boolean insert_multi_any_primary_null_boolean(
        root_primary_key_list,
        prompt_dictionary,
        multi_row_dictionary,
        row_number );

    if ( insert_multi_any_primary_null_boolean() )
    {
        continue;
    }

    INSERT_ROW *insert_row =
        insert_row_new(
            folder_name,
            root_primary_key_list,
            folder_attribute_append_isa_list,
            relation_mto1_isa_list,
            prompt_dictionary,
            multi_row_dictionary,
            ignore_name_list,
            post_change_process,
            folder_attribute_insert_required_name_list,
            row_number );

    if ( insert_row )
    {
        if ( !insert_row_list ) insert_row_list = list_new();

        list_set( insert_row_list, insert_row );
    }
}

if ( !list_length( insert_row_list ) ) this = NULL;

/* Attributes */
int dictionary_highest_index;
LIST *insert_row_list;
```

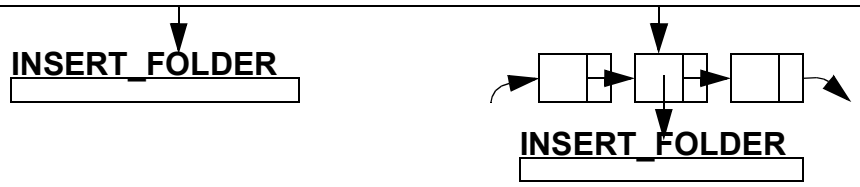


INSERT_MULTI (2)

```
/* Usage */
void insert_multi_attribute_default_set(
    LIST *folder_attribute_append_isa_list,
    DICTIONARY *multi_row_dictionary /* in/out */,
    int row_number );

/* Process */
for FOLDER_ATTRIBUTE *folder_attribute in
    folder_attribute_append_isa_list
{
    if ( folder_attribute->default_value )
    {
        void insert_multi_attribute_default_value_set(
            folder_attribute->attribute_name,
            folder_attribute->default_value,
            multi_row_dictionary /* in/out */,
            row_number );
    }
}

/* Usage */
void insert_multi_attribute_default_value_set(
    char *attribute_name,
    char *default_value,
    DICTIONARY *multi_row_dictionary /* in/out */,
    int row_number );

/* Process */
static char *insert_multi_key(
    attribute_name,
    row_number );

char *dictionary_get(
    insert_multi_key(),
    multi_row_dictionary );

if (    !dictionary_get()
||      !*dictionary_get() )
{
    dictionary_set(
        multi_row_dictionary,
        strdup( insert_multi_key() ),
        folder_attribute->default_value );
}

/* Usage */
boolean insert_multi_any_primary_null_boolean(
    LIST *root_primary_key_list,
    DICTIONARY *prompt_dictionary,
    DICTIONARY *multi_row_dictionary,
    int row_number );

/* Process */
for char *primary_key in root_primary_key_list
{
    char *dictionary_get(
        primary_key,
        prompt_dictionary );

    if (    dictionary_get()
    &&    *dictionary_get() )
    {
        continue;
    }

    static char *insert_multi_key(
        primary_key /* attribute_name */,
        row_number );

    char *dictionary_get(
        insert_multi_key(),
        multi_row_dictionary );

    if (    !dictionary_get()
    ||      !*dictionary_get() )
    {
        return 1;
    }
}

return 0;

/* Usage */
static char *insert_multi_key(
    char *attribute_name,
    int row_number );

/* Process */
```

INSERT_ROW

```
/* Usage */
INSERT_ROW *insert_row_new(
    char *folder_name,
    LIST *root_primary_key_list,
    LIST *folder_attribute_append_isa_list(),
    LIST *relation_mto1_isa_list,
    DICTIONARY *prompt_dictionary,
    DICTIONARY *multi_row_dictionary,
    LIST *ignore_name_list,
    PROCESS *post_change_process,
    LIST *folder_attribute_insert_required_name_list,
    int row_number );

/* Process */
INSERT_ROW *insert_row_calloc(
    void );

LIST *folder_attribute_name_list(
    folder_name,
    folder_attribute_append_isa_list );

INSERT_FOLDER *insert_folder =
    INSERT_FOLDER *insert_folder_new(
        folder_name,
        root_primary_key_list,
        (LIST *)0 /* isa_primary_key_list */,
        folder_attribute_append_isa_list,
        prompt_dictionary,
        multi_row_dictionary,
        ignore_name_list,
        post_change_process,
        folder_attribute_insert_required_name_list,
        row_number,
        folder_attribute_name_list() );

if ( !insert_folder ) return NULL;

if ( !list_rewind( relation_mto1_isa_list ) )
    return this;

LIST *insert_folder_isa_list = list_new();

char *appaserver_parameter_mount_point();

for RELATION_MTO1 *
    relation_mto1 in
        relation_mto1_isa_list
{
    LIST *folder_attribute_name_list(
        relation_mto1->one_folder_name,
        folder_attribute_append_isa_list );

    if ( relation_mto1->
            one_folder->
            post_change_process_name )
    {
        PROCESS *post_change_process =
            process_fetch(
                relation_mto1->
                    one_folder->
                    post_change_process_name,
                (char *)0 /* document_root */,
                (char *)0 /* relative_source_directory */,
                1 /* check_executable_inside */,
                appaserver_parameter_mount_point() );
    }
    else
    {
        post_change_process = NULL;
    }

    list_set(
        insert_folder_isa_list,
        insert_folder_new(
            relation_mto1->one_folder_name,
            root_primary_key_list,
            relation_mto1->
                one_folder->
                folder_attribute_primary_key_list
                    /* isa_primary_key_list */,
            folder_attribute_append_isa_list,
            prompt_dictionary,
            multi_row_dictionary,
            ignore_name_list,
            post_change_process,
            folder_attribute_insert_required_name_list,
            row_number,
            folder_attribute_name_list() ) );
}

/* Attribute */
INSERT_FOLDER *insert_folder;
LIST *insert_folder_isa_list;
```



INSERT_FOLDER (1)

```
/* Usage */
INSERT_FOLDER *insert_folder_new(
    char *folder_name,
    LIST *root_primary_key_list,
    LIST *isa_primary_key_list,
    LIST *folder_attribute_append_isa_list,
    DICTIONARY *prompt_dictionary,
    DICTIONARY *multi_row_dictionary,
    LIST *ignore_name_list,
    PROCESS *post_change_process,
    LIST *folder_attribute_insert_required_name_list,
    int row_number,
    LIST *folder_attribute_name_list );

/* Process */
INSERT_FOLDER *insert_folder_calloc(
    void );

static char *application_table_name(
    folder_name );

char *strdup( application_table_name() );

LIST *insert_datum_list = list_new();

if ( isa_primary_key_list )
{
    list_rewind( isa_primary_key_list );
}

for char *attribute_name in
    root_primary_key_list
{
    FOLDER_ATTRIBUTE *folder_attribute =
        folder_attribute_seek(
            (char *)0 /* folder_name */,
            attribute_name,
            folder_attribute_append_isa_list );

    boolean attribute_is_number(
        folder_attribute->
            attribute->
            datatype_name );

    INSERT_DATUM *insert_datum_extract(
        prompt_dictionary,
        multi_row_dictionary,
        row_number,
        attribute_name,
        folder_attribute->primary_key_index,
        attribute_is_number() );

    if ( !insert_datum_extract() ) return null;

    if ( isa_primary_key_list )
    {
        insert_datum_extract()->attribute_name =
            list_get( isa_primary_key_list );

        list_next( isa_primary_key_list );
    }

    list_set(
        insert_datum_list,
        insert_datum_extract() );
}

for char *attribute_name in
    folder_attribute_name_list
{
    boolean list_string_exists(
        attribute_name,
        ignore_name_list );

    if ( list_string_exists() ) return NULL;

    FOLDER_ATTRIBUTE *folder_attribute =
        folder_attribute_seek(
            (char *)0 /* folder_name */,
            attribute_name,
            folder_attribute_append_isa_list );

    if ( folder_attribute->primary_key_index )
        continue;

    boolean attribute_is_number(
        folder_attribute->
            attribute->
            datatype_name );

    INSERT_DATUM *insert_datum_extract(
        prompt_dictionary,
        multi_row_dictionary,
        row_number,
        attribute_name,
        0 /* primary_key_index */,
        attribute_is_number() );

    if ( list_length( folder_attribute_insert_required_name_list )
    &&   !insert_datum_extract() )
    {
        boolean list_string_boolean(
            attribute_name,
            folder_attribute_insert_required_name_list );

        if ( list_string_boolean() ) return null;
    }

    if ( insert_datum_extract() )
    {
        list_set(
            insert_datum_list,
            insert_datum_extract() );
    }
}
```

INSERT_FOLDER (2)

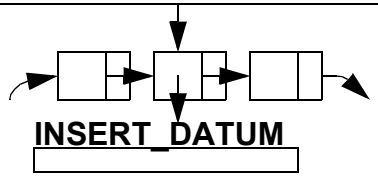
```
/* Process (continued) */
char *appaserver_table_name(
    folder_name );

char *insert_datum_sql_statement(
     appaserver_table_name(),
     insert_datum_list );

if ( post_change_process )
{
    char *post_change_process_name =
        post_change_process->process_name;

    char *post_change_process_command_line =
        post_change_process->command_line;

    LIST *insert_datum_key_datum_list(
        insert_datum_list );
}

/* Usage */
char *insert_folder_string_sql_statement(
    const char SQL_DELIMITER,
    char *table_name,
    char *attribute_name_list_string,
    char *delimited_string );
```

```
/* Process */
char *insert_folder_value_list_string(
    SQL_DELIMITER,
    delimited_striing );

char *insert_folder_sql_statement_string(
    table_name,
    attribute_name_list_string,
    insert_folder_value_list_string() )

free( insert_folder_value_list_string() );

/* Usage */
char *insert_folder_value_list_string(
    char SQL_DELIMITER,
    char *delimited_string );

/* Process */
char value_list_string[];
char *ptr = value_list_string;

for (     int p = 0;
          piece(  char datum[],
                  SQL_DELIMITER,
                  delimited_string,
                  p );
          p++ )
```

```
{
    if ( ptr != comma_string ) ptr += sprintf( ptr, "," );

    char *insert_datum_value_string(
        datum,
        0 /* not attribute_is_number */ );

    ptr += sprintf( ptr,
        "%s",
        insert_datum_value_string() );

    free( insert_datum_value_string() );
}

/* Usage */
char *insert_folder_sql_statement_string(
    char *appaserver_table_name,
    char *attribute_name_list_string,
    char *value_list_string )

/* Process */
sprintf(
    char sql_statement_string[],
    "insert into %s (%s) values (%s);",
    appaserver_table_name,
    attribute_name_list_string,
    value_list_string() );
```

INSERT_FOLDER (3)

```
/* Usage */
char *insert_folder_execute_command_line(
    char *post_change_process_command_line,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *post_change_process->process_name,
    char *appaserver_error_filespecification,
    const char *APPASERVER_INSERT_STATE,
    LIST *insert_datum_list,
    LIST *insert_datum_key_datum_list,
    int row_number,
    int row_count );

/* Process */
char *process_replace_command_line(
    UPDATE_PREUPDATE_PREFIX,
    (char *)0 /* application_name */,
    session_key,
    login_name,
    role_name,
    folder_name,
    (char *)0 /* target_frame */,
    APPASERVER_INSERT_STATE,
    process_name,
    (char *)0 /* many_folder_name */,
    (char *)0 /* one_folder_name */,
    (char *)0 /* related_column */,
    (char *)0 /* update_results_string */,
    (char *)0 /* update_error_string */,
    (DICTIONARY *)0 /* operation_row_list_dictionary */,
    (DICTIONARY *)0 /* dictionary_single_row */,
    (char *)0 /* where_string */,
    row_number,
    row_count,
    0 /* parent_process_id */,
    insert_datum_key_datum_list /* primary_key_data_list */,
    "y" /* execute_yn */,
    insert_datum_list,
    (LIST *)0 /* update_attribute_list */,
    (LIST *)0 /* query_cell_list */,
    appaserver_error_filespecification,
    post_change_process_command_line );

/* Attributes */
char *application_table_name;
LIST *insert_datum_list;
char *insert_datum_sql_statement;
char *post_change_process_name;
char *post_change_process_command_line;
LIST *insert_datum_key_datum_list;

/* Set externally */
char *execute_command_line;
```



INSERT_DATUM (1)

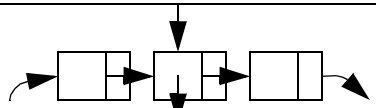
```
/* Usage */
INSERT_DATUM *insert_datum_extract(
    DICTIONARY *prompt_dictionary,
    DICTIONARY *multi_row_dictionary,
    int row_number,
    char *attribute_name,
    int primary_key_index,
    boolean attribute_is_number );

/* Process */
char *datum =
    dictionary_get(
        attribute_name,
        prompt_dictionary );

if ( datum && *datum )
{
    return
    insert_datum_new(
        attribute_name,
        datum,
        primary_key_index,
        attribute_is_number );
}

if ( multi_row_dictionary )
{
    char *widget_container_key(
        atribute_name /* widget_name */,
        row_number );

    char *datum =
        dictionary_get(
            widget_container_key(),
            multi_row_dictionary );

    if ( datum && *datum )
    {
        return
        insert_datum_new(
            attribute_name,
            datum,
            primary_key_index,
            attribute_is_number );
    }
}

return NULL;

/* Usage */
INSERT_DATUM *insert_datum_new(
    char *attribute_name,
    char *datum,
    int primary_key_index,
    boolean attribute_is_number );

/* Process */
INSERT_DATUM *insert_datum_calloc( void );

/* Usage */
char *insert_datum_sql_statement(
    char *appaserver_table_name,
    LIST *insert_datum_list );

/* Process */
char *insert_datum_attribute_name_list_string(
    insert_datum_list );

char *insert_datum_value_list_string(
    insert_datum_list );

char *insert_folder_sql_statement_string(
    appaserver_table_name,
    insert_datum_attribute_name_list_string(),
    insert_datum_value_list_string() );

free( insert_datum_attribute_name_list_string() );
free( insert_datum_value_list_string() );
```

INSERT_DATUM (2)

```
/* Usage */
char *insert_datum_attribute_name_list_string(
    LIST *insert_datum_list );

/* Process */
char string[];
char *ptr = string;

for INSERT_DATUM *insert_datum in
    insert_datum_list
{
    if ( ptr != string ) ptr += sprintf( ptr, "," );

    ptr += sprintf( ptr,
        "%s",
        insert_datum->attribute_name );
}

/* Usage */
char *insert_datum_value_list_string(
    LIST *insert_datum_list );

/* Process */
char string[];
char *ptr = string;

for INSERT_DATUM *insert_datum in
    insert_datum_list
{
    if ( ptr != string ) ptr += sprintf( ptr, "," );

    char *insert_datum_value_string(
        insert_datum->datum,
        insert_datum->attribute_is_number );

    ptr += sprintf( ptr,
        "%s",
        insert_datum_value_string() );

    free( insert_datum_value_string() );
}

/* Usage */
char *insert_datum_value_string(
    char *datum,
    boolean attribute_is_number );

/* Process */
if ( !datum || !*datum )
    strcpy( value_string, "null" );
else
if ( attribute_is_number )
    sprintf( value_string[], "%s", datum );
else
{
    char *string_escape_quote(
        char destination[],
        datum /* source */ );

    snprintf(
        value_string[],
        sizeof( value_string ),
        "'%s'",
        string_escape_quote() );
}

/* Usage */
LIST *insert_datum_key_datum_list(
    LIST *insert_datum_list );

/* Process */
LIST *key_datum_list = list_new();

for INSERT_DATUM *insert_datum in
    insert_datum_list
{
    if ( !insert_datum->primary_key_index )
    {
        return key_datum_list;
    }

    list_set( key_datum_list, insert_datum->datum );
}

/* Attributes */
char *attribute_name;
char *datum;
int primary_key_index,
boolean attribute_is_number;
```

INSERT_STATEMENT (1)

```
/* Usage */
INSERT_STATEMENT *insert_statement_new(
    char *session_key,
    char *login_name,
    char *role_name,
    char *appaserver_error_filespecification,
    INSERT_ZERO *insert_zero,
    INSERT_MULTI *insert_multi );

/* Process */
if ( !insert_zero &&!insert_multi ) exit( 1 );

INSERT_STATEMENT *insert_statement_calloc( void );

if ( insert_zero )
{
    LIST *insert_folder_statement_list = list_new();

    list_set(
        insert_folder_statement_list,
        insert_folder_statement_new(
            session_key,
            login_name,
            role_name,
            appaserver_error_filespecification,
            insert_zero->insert_folder,
            insert_zero->insert_folder_isa_list,
            1 /* row_number */,
            1 /* row_count */ ) );
}
else
{
    LIST *insert_folder_statement_list =
        session_key,
        login_name,
        role_name,
        appaserver_error_filespecification,
        insert_statement_multi_list(
            insert_multi->insert_row_list );
}

if ( !list_length( insert_folder_statement_list ) ) exit( 1 );

/* Usage */
/* Returns insert_folder_statement_list */
LIST *insert_statement_multi_list(
    char *session_key,
    char *login_name,
    char *role_name,
    char *appaserver_error_filespecification,
    LIST *insert_row_list );

/* Process */
int insert_statement_row_count(
    LIST *insert_row_list );

for INSERT_ROW *insert_row in
    insert_row_list
{
    list_set(
        insert_folder_statment_list,
        insert_folder_statement_new(
            session_key,
            login_name,
            role_name,
            appaserver_error_filespecification,
            insert_row->insert_folder,
            insert_row->insert_folder_isa_list,
            ++row_number,
            insert_statement_row_count() );
}

/* Attribute */
LIST *insert_folder_statement_list;
```



INSERT_FOLDER_STATEMENT

LIST *sql_statement_list
LIST *command_line_list

INSERT_STATEMENT (2)

```
/* Driver */
/* Returns insert_statement_error_string or null */
char *insert_statement_execute(
    char *application_name,
    LIST *insert->
          insert_statement->
          insert_folder_statement_list,
    char *insert->
          appaserver_error_filespecification );

/* Process */
LIST *insert_statement_extract_sql_list(
    LIST *insert_folder_statement_list );

if ( !list_length( insert_statement_extract_sql_list() )
        exit( 1 );

char *error_string =
    char *sql_execute(
        appaserver_error_filespecification,
        insert_statement_extract_sql_list()
            /* insert_sql_list */,
        (char *)0 /* insert_statement */ );

LIST *insert_statement_extract_command_list(
    LIST *insert_folder_statement_list );

void insert_statement_command_execute(
    insert_statement_extract_command_list() );

if ( error_string )
{
    appaserver_error_message(
        application_name,
        (char *)0 /* login_name */,
        error_string );
    return error_string;
}

/* Usage */
void insert_statement_command_execute(
    LIST *insert_statement_extract_command_list() );

/* Process */
for char *command in
    insert_statement_extract_command_list
{
    security_system(
        SECURITY_FORK_CHARACTER,
        SECURITY_FORK_STRING,
        command );
}
```

INSERT_FOLDER_STATEMENT

```
/* Usage */
INSERT_FOLDER_STATEMENT *
    insert_folder_statement_new(
        char *session_key,
        char *login_name,
        char *role_name,
        char *appaserver_error_filespecification,
        INSERT_FOLDER *insert_folder,
        LIST *insert_folder_isa_list,
        int row_number,
        int row_count );

/* Process */
INSERT_FOLDER_STATEMENT *
    insert_folder_sql_statement_calloc(
        void );

LIST *sql_statement_list = list_new();

list_set(
    sql_statement_list,
    insert_folder->insert_datum_sql_statement );

if ( insert_folder->post_change_process_command_line )
{
    command_line_list = list_new();

    char *insert_folder_execute_command_line(
        insert_folder->post_change_process_command_line,
        session_key,
        login_name,
        role_name,
        insert_folder->application_table_name
            /* folder_name */,
        insert_folder->post_change_process_name,
        appaserver_error_filespecification,
        APPASERVER_INSERT_STATE,
        insert_folder->insert_datum_list,
        insert_folder->insert_datum_key_datum_list,
        row_number,
        row_count );

    list_set(
        command_line_list,
        insert_folder_execute_ommand_line() );
}

for INSERT_FOLDER *insert_folder in
    insert_folder_isa_list
{
    list_set(
        sql_statement_list,
        insert_folder->insert_datum_sql_statement );

    if ( insert_folder->post_change_process_command_line )
    {
        if ( !command_line_list )
            command_line_list =
                list_new();

        char *insert_folder_execute_command_line(
            insert_folder->post_change_process_command_line,
            session_key,
            login_name,
            role_name,
            insert_folder->application_table_name
                /* folder_name */,
            insert_folder->post_change_process_name,
            appaserver_error_filespecification,
            APPASERVER_INSERT_STATE,
            insert_folder->insert_datum_list,
            insert_folder->insert_datum_key_datum_list,
            row_number,
            row_count );

        list_set(
            command_line_list,
            insert_folder_execute_ommand_line() );
    }
}

/* Usage */
int insert_folder_statement_sql_list_length(
    LIST *insert_folder_statement_list );

/* Process */
int length = 0;

for INSERT_FOLDER_STATEMENT *
    insert_folder_statement in
        insert_folder_statement_list
{
    length +=
        list_length(
            insert_folder_statement->
                sql_statement_list );
}

/* Usage */
char *insert_folder_statement_system_string(
    const char SQL_DELIMITER,
    char *application_table_name(),
    char *field_list_string );

/* Process */
snprintf(
    system_string[],
    sizeof( system_string ),
    "insert_statement.e "
    "table_name=%s "
    "field_list=%s "
    "delimiter='%c'",
    application_table_name,
    field_list_string,
    SQL_DELIMITER );

/* Attributes */
LIST *sql_statement_list;
LIST *command_line_list;
```

OPTIONAL_COLUMN

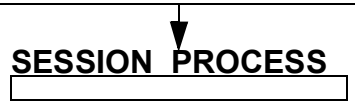
```
/* Usage */
OPTIONAL_COLUMN *optional_column_new(
    const char delimiter,
    char *base_string,
    char *component /* column or datum (stack memory) */,
    boolean escape_boolean,
    boolean set_boolean );

/* Process */
OPTIONAL_COLUMN *optional_column_calloc(
    void );

if ( !set_boolean )
{
    return_string = strdup( base_string );
}
else
{
    if ( escape_boolean )
    {
        char *component =
            security_sql_injection_escape(
                SECURITY_ESCAPE_CHARACTER_STRING,
                component );
    }

    char *optional_column_return_string(
        delimiter,
        base_string,
        component );

    if ( escape_boolean ) free( component );
}

char *prior_return_string = base_string;

/* Usage */
char *optional_column_return_string(
    const char delimiter,
    char *base_string,
    char *component );

/* Process */
snprintf(
    return_string[],
    sizeof ( return_string ),
    "%s%c%s",
    base_string,
    delimiter,
    (component) ? component : "" );

/* Usage */
OPTIONAL_COLUMN *optional_column_money_new(
    const char delimiter,
    char *base_string,
    double money,
    boolean set_boolean );

/* Process */
snprintf(
    component[],
    sizeof ( component ),
    "%.2lf",
    money );

OPTIONAL_COLUMN *optional_column_new(
    delimiter,
    base_string,
    component /* column or datum */,
    0 /* not escape_boolean */,
    set_boolean );

/* Usage */
OPTIONAL_COLUMN *optional_column_integer_new(
    const char delimiter,
    char *base_string,
    int integer,
    boolean set_boolean );

/* Process */
snprintf(
    component[],
    sizeof ( component ),
    "%d",
    integer );

OPTIONAL_COLUMN *optional_column_new(
    delimiter,
    base_string,
    component /* column or datum */,
    0 /* not escape_boolean */,
    set_boolean );

/* Usage */
OPTIONAL_COLUMN *optional_column_text_new(
    const char delimiter,
    char *base_string,
    char *text,
    boolean set_boolean );

/* Process */
OPTIONAL_COLUMN *optional_column_new(
    delimiter,
    base_string,
    text /* column or datum */,
    0 /* not escape_boolean */,
    set_boolean );

/* Attributes */
char *return_string /* heap memory */;
char *prior_return_string;
```

SQL

```
/* Usage */

/* Returns error_string or null */
char *sql_execute(
    char *appaserver_error_filename,
    LIST *sql_list /* mutually exclusive */,
    char *sql_statement /* mutually_exclusive */ );

/* Process */
char *timlib_temp_filename( “insert” /* key */ );

snprintf(
    system_string[],
    sizeof ( system_string ),
    “tee -a %s | sql.e 2>%s”,
    appaserver_error_filename,
    timlib_temp_filename() );

FILE *appaserver_output_pipe(
    system_string );

if ( sql_statement )
{
    fprintf(
        appaserver_output_pipe(),
        “%s\n”,
        sql_statement );
}
else
for char *sql_statement in sql_list
{
    fprintf(
        appaserver_output_pipe(),
        “%s\n”,
        sql_statement );
}

pclose( appaserver_output_pipe() );

char *error_string =
    string_file_fetch(
        timlib_temp_filename(),
        “\n<br>” /* delimiter */ );

timlib_remove_file( timlib_temp_filename() );

char *string_search_replace(
    error_string,
    “ERROR 1062 (23000) “,
    ““ );
```

POST_CHOOSE_PROCESS (1)

```
/* Usage */
POST_CHOOSE_PROCESS *
    post_choose_process_new(
        int argc,
        char **argv,
        char *application_name,
        char *session_key,
        char *login_name,
        char *role_name,
        char *process_name );

/* Process */
/* Sets appaserver environment and outputs argv. */
/* Each parameter is security inspected. */
/* Any error will exit( 1 ). */
SESSION_PROCESS *session_process =
    session_process_integrity_exit(
        argc,
        argv,
        application_name,
        session_key,
        login_name,
        role_name,
        process_name );

POST_CHOOSE_PROCESS *
    post_choose_process_calloc(
        void );

if ( session_process->process_name )
{
    boolean post_choose_process_no_parameters(
        char *session_process->process_name );
}

if ( post_choose_process_no_parameters() )
{
    char *post_prompt_process_system_string(
        POST_PROMPT_PROCESS_EXECUTABLE,
        application_name,
        session_key,
        login_name,
        role_name,
        session_process->process_name );
}
else
{
    boolean process_parameter_drillthru_boolean(
        session_process->process_name );

    char *prompt_process_output_system_string(
        PROMPT_PROCESS_OUTPUT_EXECUTABLE,
        session_key,
        login_name,
        role_name,
        process_name,
        (char *)0 /* dictionary_separate_send_string() */,
        process_parameter_drillthru_boolean(),
        process_parameter_drillthru_boolean()
            /* is_drillthru */,
        appaserver_error_filename() );
}

/* Attributes */
SESSION_PROCESS *session_process;
boolean no_parameters;
boolean process_parameter_drillthru_boolean;
char *post_prompt_process_system_string;
char *prompt_process_output_system_string;
```



POST_CHOOSE_PROCESS (2)

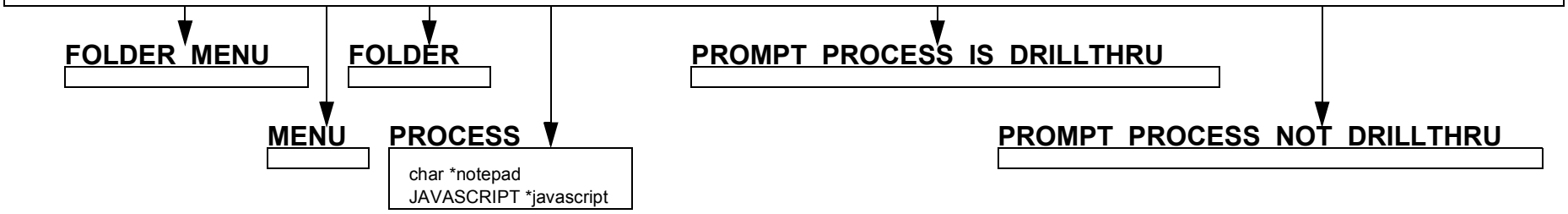
```
/* Usage */
char *post_choose_process_action_string(
    char *POST_CHOOSE_PROCESS_EXECUTABLE,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name );

/* Process */
static action_string[];

if ( *action_string ) return action_string;

char *appaserver_parameter_cgi_directory();

char *application_server_address();

char *application_ssl_support_boolean(
    application_name );

char *application_http_prompt(
    appaserver_parameter_cgi_directory(),
    appliction_server_address(),
    application_ssl_support_boolean() );

snprintf(
    static action_string[],
    sizeof ( action_string ),
    "%s/%s?%s+%s+%s+%s",
    application_http_prompt(),
    POST_CHOOSE_PROCESS_EXECUTABLE,
    application_name,
    session_key,
    login_name,
    role_name );

/* Driver */
if ( post_prompt_process_system_string )
{
    system( post_prompt_process_system_string );
}
else
if ( prompt_process_output_system_string )
{
    system( prompt_process_output_system_string );
}
```

PROMPT_PROCESS (1)

```
/* Usage */
PROMPT_PROCESS *prompt_process_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *process_name,
    boolean has_drillthru,
    boolean is_drillthru,
    POST_DICTIONARY *post_dictionary,
    boolean menu_horizontal_boolean(),
    char *appaserver_parameter->data_directory,
    char *appaserver_parameter->document_root,
    char *appaserver_parameter->mount_point,
    char *application_relative_source_directory() );

/* Process */
PROMPT_PROCESS *prompt_process_calloc(
    void );

if ( menu_horizontal_boolean )
{
    FOLDER_MENU *folder_menu =
        folder_menu_fetch(
            application_name,
            session_key,
            role_name,
            data_directory,
            0 /* not folder_menu_remove_boolean */ );

    MENU *menu =
        menu_new(
            application_name,
            session_key,
            login_name,
            role_name,
            1 /* application_menu_horizontal_boolean */,
            folder_menu->count_list );
}

PROCESS *process =
    process_fetch(
        process_name,
        document_root,
        application_relative_source_directory,
        1 /* check_executable_inside */,
        mount_point );

LIST *prompt_process_javascript_filename_list(
    PROCESS *process );

if ( is_drillthru )
{
    PROMPT_PROCESS_IS_DRILLTHRU *
        prompt_process_is_drillthru =
            prompt_process_is_drillthru_new(
                application_name,
                session_key,
                login_name,
                role_name,
                process_name,
                process,
                menu,
                prompt_process_javascript_filename_list() );
}
else
{
    PROMPT_PROCESS_NOT_DRILLTHRU *
        prompt_process_not_drillthru =
            prompt_process_not_drillthru_new(
                application_name,
                session_key,
                login_name,
                role_name,
                process_name,
                process,
                has_drillthru,
                post_dictionary,
                menu,
                prompt_process_javascript_filename_list() );
}

char *prompt_process_html(
    PROMPT_PROCESS_IS_DRILLTHRU *
        prompt_process_is_drillthru,
    PROMPT_PROCESS_NOT_DRILLTHRU *
        prompt_process_not_drillthru );
```

PROMPT_PROCESS (2)

```
/* Usage */
char *prompt_process_notepad(
    PROCESS *process );

/* Process */
char *process->notepad;

/* Usage */
char *prompt_process_output_system_string(
    char *PROMPT_PROCESS_OUTPUT_EXECUTABLE,
    char *session_key,
    char *login_name,
    char *role_name,
    char *process_name,
    char *dictionary_separate_send_string(),
    boolean has_drillthru,
    boolean is_drillthru,
    char *appaserver_error_filename() );

/* Process */

/* Driver */
printf( "%s\n", prompt_process->html );

/* Attributes */
FOLDER_MENU *folder_menu;
MENU *menu;
char *process_name;
PROCESS *process;
LIST *javascript_filename_list;
PROMPT_PROCESS_IS_DRILLTHRU *
    prompt_process_is_drillthru;
PROMPT_PROCESS_NOT_DRILLTHRU *
    prompt_process_not_drillthru;
char *html;
```

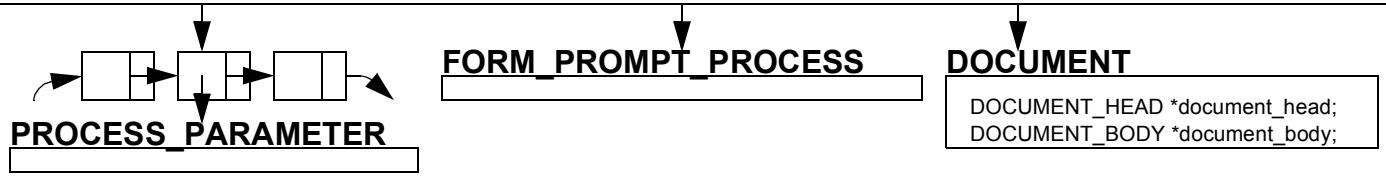


PROMPT_PROCESS_IS_DRILLTHRU (1)

```
/* Usage */
PROMPT_PROCESS_IS_DRILLTHRU *
    prompt_process_is_drillthru_new(
        char *application_name,
        char *session_key,
        char *login_name,
        char *role_name,
        char *process_name,
        PROCESS *process,
        MENU *menu,
        LIST *prompt_process_javascript_filename_list() );

/* Process */
PROMPT_PROCESS_IS_DRILLTHRU *
    prompt_process_is_drillthru_calloc(
        void );

char *process_parameter_where(
        process_name,
        1 /* is_drillthru */ );

char *process_parameter_system_string(
    PROCESS_PARAMETER_SELECT,
    PROCESS_PARAMETER_TABLE,
    process_parameter_where();

LIST *process_parameter_list =
    process_parameter_system_list(
        application_name,
        login_name,
        role_name,
        (DICTIONARY *)0 /* drillthru_dictionary */,
        process_parameter_system_string() );

char *post_change_javascript =
    process->post_change_javascript;

if ( menu )
{
    char *document_head_menu_setup_string(
        1 /* menu_boolean */ );

    char *document_head_calendar_setup_string(
        process_parameter_date_boolean(
            process_parameter_list ) );
}

char *prompt_process_notepad(
    process );

FORM_PROMPT_PROCESS *form_prompt_process =
    form_prompt_process_new(
        application_name,
        session_key,
        login_name,
        role_name,
        process_name,
        process_parameter_list,
        post_change_javascript,
        1 /* is_drillthru */ );

char *prompt_process_is_drillthru_title_string(
    char *process_name );

char *javascript_include_string(
    application_name,
    prompt_process_javascript_filename_list );

char *onload_javascript_string =
    document_body_set_onload_javascript_string(
        /* May free() first parameter */
        form_prompt_process->
            recall_onload->
            javascript
                /* javascript_string */,
        post_change_javascript
                /* append_javascript_string */ );

DOCUMENT *document =
    document_new(
        application_name,
        application_title_string( application_name ),
        prompt_process_is_drillthru_title_string(),
        (char *)0 /* sub_title_string */,
        (char *)0 /* sub_sub_title_string */,
        prompt_process_notepad(),
        onload_javascript_string,
        (menu) ? menu->html : (char *)0,
        document_head_menu_setup_string(),
        document_head_calendar_setup_string(),
        javascript_include_string() );

char *document_form_html(
    document->html,
    document->document_head->html,
    document_head_close_tag(),
    document->document_body->html,
    form_prompt_process->form_prompt_lookup_html,
    document_body_close_tag(),
    document_close_tag() );

free( form_prompt_process->
        form_prompt_lookup_html );
```

PROMPT_PROCESS_IS_DRILLTHRU (2)

```
/* Attributes */
LIST *process_parameter_list;
char *post_change_javascript;
char *document_head_menu_setup_string;
char *document_head_calendar_setup_string;
char *prompt_process_notepad;
FORM_PROMPT_PROCESS *form_prompt_process;
char *title_string;
char *javascript_include_string;
char *onload_javascript_string;
DOCUMENT *document;
char *document_form_html;
```



PROCESS_PARAMETER

FORM_PROMPT_PROCESS

DOCUMENT

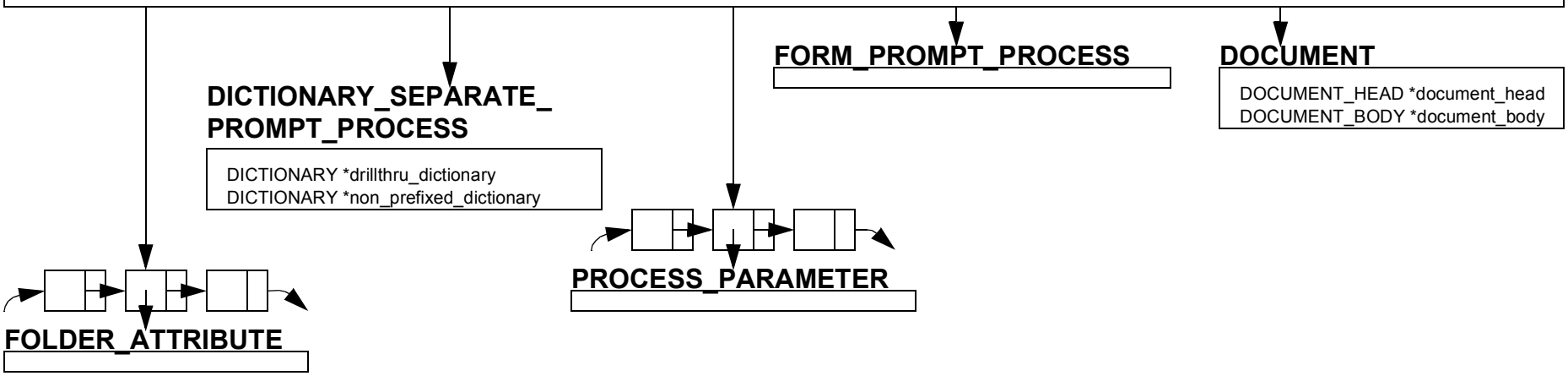
```
DOCUMENT_HEAD *document_head;
DOCUMENT_BODY *document_body;
```

PROMPT_PROCESS_NOT_DRILLTHRU (1)

```
/* Usage */
PROMPT_PROCESS_NOT_DRILLTHRU *
    prompt_process_not_drillthru_new(
        char *application_name,
        char *session_key,
        char *login_name,
        char *role_name,
        char *process_name,
        PROCESS *process,
        boolean has_drillthru,
        POST_DICTIONARY *post_dictionary,
        MENU *menu,
        LIST *prompt_process_javascript_filename_list() );

/* Process */
DICTIONARY *drillthru_dictionary = {0};

if ( has_drillthru )
{
    LIST *process_parameter_list(
        process_name,
        0 /* not drillthru_boolean */ );

LIST *process_parameter_folder_name_list(
    process_parameter_list() );

if ( list_length(
        process_parameter_folder_name_list() )
{
    LIST *folder_attribute_name_list_attribute_list(
        process_parameter_folder_name_list() );
}

LIST *process_parameter_date_name_list(
    process_parameter_list() );

DICTIONARY_SEPARATE_PROMPT_PROCESS *
    dictionary_separate_prompt_process =
        dictionary_separate_prompt_process_new(
            post_dictionary->original_post_dictionary,
            application_name,
            login_name,
            process_parameter_date_name_list(),
            folder_attribute_name_list_attribute_list() );

drillthru_dictionary =
        dictionary_separate_prompt_process->
            drillthru_dictionary;
}

char *process_parameter_where(
    process_name,
    0 /* not is_drillthru */ );

char *process_parameter_system_string(
    PROCESS_PARAMETER_SELECT,
    PROCESS_PARAMETER_TABLE,
    process_parameter_where() );

LIST *process_parameter_list =
    process_parameter_system_list(
        application_name,
        login_name,
        role_name,
        drillthru_dictionary,
        process_parameter_system_string() );

char *post_change_javascript =
    process->post_change_javascript;
```

PROMPT_PROCESS_NOT_DRILLTHRU (2)

```
/* Process (continued) */
if ( menu )
{
    char *document_head_menu_setup_string(
        1 /* menu_boolean */ );

    char *document_head_calendar_setup_string(
        process_parameter_date_boolean(
            process_parameter_list ) );
}

char *prompt_process_notepad(
    process,
    process_set );

FORM_PROMPT_PROCESS *form_prompt_process =
    form_prompt_process_new(
        application_name,
        session_key,
        login_name,
        role_name,
        process_name,
        process_parameter_list,
        post_change_javascript,
        0 /* not is_drillthru */ );

char *prompt_process_not_drillthru_title_string(
    char *process_name );

char *javascript_include_string(
    application_name,
    prompt_process_javascript_filename_list );

char *onload_javascript_string =
    document_body_set_onload_javascript_string(
        form_prompt_process->
            recall_onload->
            javascript
                /* javascript_string */,
        post_change_javascript
                /* append_javascript_string */ );

DOCUMENT *document =
    document_new(
        application_name,
        application_title( application_name ),
        prompt_process_not_drillthru_title_string(),
        (char *)0 /* sub_title_string */,
        (char *)0 /* sub_sub_title_string */,
        prompt_process_notepad(),
        onload_javascript_string,
        (menu) ? menu->html : (char *)0,
        document_head_menu_setup_string(),
        document_head_calendar_setup_string(),
        javascript_include_string() );

char *document_form_html(
    document->html,
    document->document_head->html,
    document_head_close_tag(),
    document->document_body->html,
    form_prompt_process->form_prompt_lookup_html,
    document_body_close_tag(),
    document_close_tag() );

free( form_prompt_process->
          form_prompt_lookup_html );

/* Attributes */
LIST *process_parameter_folder_name_list;
LIST *folder_attribute_name_list_attribute_list;
LIST *process_parameter_date_name_list;
DICTIONARY_SEPARATE_PROMPT_PROCESS *
    dictionary_separate_prompt_process;
char *process_parameter_where;
char *process_parameter_system_string;
char *process_parameter_set_where;
char *process_parameter_set_system_string;
LIST *process_parameter_list;
char *post_change_javascript;
char *document_head_menu_setup_string;
char *document_head_calendar_setup_string;
char *prompt_process_notepad;
FORM_PROMPT_PROCESS *form_prompt_process;
char *title_string;
char *javascript_include_string;
char *onload_javascript_string;
DOCUMENT *document;
char *document_form_html;
```



POST_PROMPT_PROCESS (1)

```
/* Usage */
POST_PROMPT_PROCESS *
   post_prompt_process_new(
      int argc,
      char **argv,
      char *application_name,
      char *session_key,
      char *login_name,
      char *role_name,
      char *process_name,
      boolean is_drillthru );

/* Process */
POST_PROMPT_PROCESS *
   post_prompt_process_calloc(
      void );

/* Sets appaserver environment and outputs argv. */
/* Each parameter is security inspected. */
/* Any error will exit( 1 ). */
SESSION_PROCESS *session_process =
   session_process_integrity_exit(
      argc,
      argv,
      application_name,
      session_key,
      login_name,
      role_name,
      process_name );

boolean application_menu_horizontal_boolean(
   application_name );

boolean menu_horizontal_boolean(
   FRAMESET_PROMPT_FRAME /* target_frame */,
   FRAMESET_PROMPT_FRAME,
   application_menu_horizontal_boolean() );

APPASERVER_PARAMETER *
   appaserver_parameter =
      appaserver_parameter_new();

LIST *process_parameter_list(
   session_process->process_name,
   is_drillthru );

LIST *process_parameter_upload_filename_list(
   process_parameter_list() );

POST_DICTIONARY *post_dictionary =
   post_dictionary_stdin_new(
      WIDGET_UPLOAD_RECALL_PREFIX,
      application_name,
      appaserver_parameter->upload_directory,
      process_parameter_upload_filename_list() );

LIST *process_parameter_date_name_list(
   process_parameter_list() );

LIST *process_parameter_folder_name_list(
   process_parameter_list() );

LIST *folder_attribute_name_list_attribute_list(
   process_parameter_folder_name_list() );

DICTIONARY_SEPARATE_PROMPT_PROCESS *
   dictionary_separate_prompt_process =
      dictionary_separate_prompt_process_new(
         post_dictionary->original_post_dictionary,
         application_name,
         login_name,
         prompt_process_date_name_list(),
         folder_attribute_name_list_attribute_list() );

if ( is_drillthru )
{
   char *prompt_process_output_system_string(
      PROMPT_PROCESS_OUTPUT_EXECUTABLE,
      session_key,
      login_name,
      role_name,
      process_session->process_name,
      dictionary_separate_send_string(),
      1 /* has_drillthru */,
      0 /* not is_drillthru */,
      appaserver_error_filename() );

   return post_prompt_process;
}

char *application_relative_source_directory(
   application_name );

PROCESS *process =
   process_fetch(
      session_process->process_name,
      appaserver_parameter->document_root,
      application_relative_source_directory(),
      1 /* check_executable_inside */,
      appaserver_parameter->mount_point );

if ( !process ) exit( 1 );
```

SESSION_PROCESS

POST_DICTIONARY

PROCESS

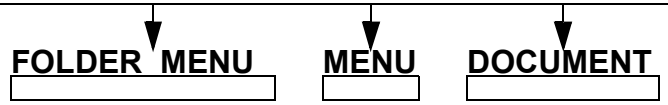
APPASERVER_PARAMETER

DICTIONARY_SEPARATE_PROMPT_PROCESS

DICTIONARY *drillthru_dictionary
DICTIONARY *non_prefixed_dictionary

POST_PROMPT_PROCESS (2)

```
/* Process (continued) */
char *appaserver_error_filespecification(
    application_name );

char *post_prompt_process_command_line(
    process->command_line,
    application_name,
    session_key,
    login_name,
    role_name,
    session_process->process_name,
    dictionary_separate->non_prefixed_dictionary
        /* in/out */,
    appaserver_error_filespecification() );

boolean post_choose_process_no_parameters(
    session_process->process_name );

if ( post_choose_process_no_parameters() )
{
    if ( menu_horizontal_boolean() )
    {
        FOLDER_MENU *folder_menu =
            folder_menu_fetch(
                application_name,
                session_key,
                role_name,
                appaserver_parameter->data_directory,
            0 /* not folder_menu_remove_boolean */ );

    MENU *menu =
        menu_new(
            application_name,
            session_key,
            login_name,
            role_name,
            1 /* application_menu_horizontal_boolean */,
            folder_menu->count_list );
}

DOCUMENT *document = document_new(
    application_name,
    application_title_string( application_name ),
    (char *)0 /* title_string */,
    (char *)0 /* sub_title_string */,
    (char *)0 /* sub_sub_title_string */,
    (char *)0 /* notepad */,
    (char *)0 /* onload_javascript_string */,
    (menu) ? menu->html : (char *)0,
    document_head_menu_setup_string(
        (menu) ? 1 : 0 /* menu_boolean */ ),
    (char *)0 /* calendar_setup_string */,
    javascript_include_string(
        application_name,
        (LIST *)0 /* javascript_filename_list */ ) );

char *document_form_html(
    document->html,
    document->document_head->html,
    document_head_close_tag(),
        document->document_body->html,
        (char *)0 /* form_html */,
        (char *)0 /* document_body_close_tag */,
        (char *)0 /* document_close_tag */ );
}

/* Attributes */
SESSION_PROCESS *session_process;
boolean application_menu_horizontal_boolean;
boolean menu_horizontal_boolean;
APPASERVER_PARAMETER *appaserver_parameter;
LIST *process_parameter_list;
LIST *process_parameter_upload_filename_list;
POST_DICTIONARY *post_dictionary;
LIST *prompt_process_date_name_list;
LIST *process_parameter_folder_name_list;
LIST *folder_attribute_name_list_attribute_list;
DICTIONARY_SEPARATE_PROMPT_PROCESS *
    dictionary_separate_prompt_process;
char *prompt_process_output_system_string;
char *application_relative_source_directory;
PROCESS *process;
char *command_line;
boolean post_choose_process_no_parameters;
FOLDER_MENU *folder_menu;
MENU *menu;
DOCUMENT *document;
char *document_form_html;
```

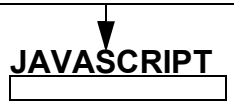


POST_PROMPT_PROCESS (3)

```
/* Usage */
char *post_prompt_process_command_line(
    char *process_command_line,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *session_process->process_name,
    DICTIONARY *dictionary_separate->
       non_prefixed_dictionary /* in/out */,
    char *appaserver_error_filenspecification );

/* Process */
DICTIONARY *
    post_prompt_process_non_prefixed_dictionary(
       const char *
          PROCESS_FOLDER_NAME_PLACEHOLDER,
       const char *
          PROCESS_TABLE_NAME_PLACEHOLDER,
       const char *PROCESS_ATTRIBUTE_PLACEHOLDER,
       const char *PROCESS_COLUMN_PLACEHOLDER,
       DICTIONARY *non_prefixed_dictionary /* in/out */ );

char *execute_yn =
    dictionary_get(
       APPASERVER_EXECUTE_YN,
       post_prompt_process_non_prefixed_dictionary() );

char *process_replace_command_line(
    UPDATE_PREUPDATE_PREFIX,
    application_name,
    session_key,
    login_name,
    role_name,
    (char *)0 /* folder_name */,
    (char *)0 /* target_frame */,
    (char *)0 /* state */,
    process_name,
    (char *)0 /* many_folder_name */,
    (char *)0 /* one_folder_name */,
    (char *)0 /* related_column */,
    (char *)0 /* update_results_string */,
    (char *)0 /* update_error_string */,
    post_prompt_process_non_prefixed_dictionary()
       /* operation_row_list_dictionary */,
    post_prompt_process_non_prefixed_dictionary
       /* dictionary_single_row */,
    (char *)0 /* where_string */,
    0 /* row_number */,
    0 /* row_count */,
    0 /* parent_process_id */,
    (LIST *)0 /* primary_key_data_list */,
    execute_yn,
    (LIST *)0 /* insert_datum_list */,
    (LIST *)0 /* update_attribute_list */,
    (LIST *)0 /* query_cell_list */,
    appaserver_error_filespecification,
    process_command_line );

/* Usage */
char *post_prompt_process_system_string(
    char *POST_PROMPT_PROCESS_EXECUTABLE,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *process_name,
    char *appaserver_error_filename() );

/* Process */
/* Driver */
if ( post_prompt_process->
       prompt_process_output_system_string )
{
    security_system(
       SECURITY_FORK_CHARACTER,
       SECURITY_FORK_STRING,
       post_prompt_process->
          prompt_process_output_system_string );
}
else
if ( post_prompt_process->command_line )
{
    if ( post_prompt_process->
          post_choose_process_no_parameters )
    {
       printf(
          "%s\n",
          post_prompt_process->
             document_form_html );

       fflush( stdout );
    }

    appaserver_error_message_file(
       application_name,
       login_name,
       post_prompt_process->command_line );

    security_system(
       SECURITY_FORK_CHARACTER,
       SECURITY_FORK_STRING,
       post_prompt_process->command_line )
}
```

PROCESS (1)

```
/* Usage */
PROCESS *process_fetch(
    char *process_name,
    /* If fetching javascript */
    char *application_parameter->document_root,
    char *application_relative_source_directory(),
    /* If preparing to execute */
    boolean check_executable_inside_filesystem,
    char *appaserver_parameter->mount_point );

/* Process */
static LIST *process_list = {0};

if ( !process_list ) process_list =
{
    process_fetch_list();
}

PROCESS *process =
    process_seek(
        process_name,
        process_list );

if ( !process ) exit( 1 );

if (    process->javascript_filename
&&  document_root
&&  application_relative_source_directory )
{
    JAVASCRIPT *javascript =
        javascript_new(
            process->javascript_filename,
            document_root,
            application_relative_source_directory );
}

if ( check_executable_inside_filesystem )
{
    boolean process_executable_okay(
        mount_point,
        process->command_line );
    if ( !process_executable_okay() ) exit( 1 );
}

/* Usage */
LIST *process_fetch_list(
    void );

/* Process */
char *appaserver_system_string(
    PROCESS_SELECT,
    PROCESS_TABLE,
    (char *)0 /* where */ );

LIST *process_system_list(
    appaserver_system_string() );

/* Usage */
LIST *process_system_list(
    char *appaserver_system_string() );

/* Process */
LIST *list = list_new();

FILE *appaserver_input_pipe(
    appaserver_system_string );

for char input[] in string_input( appaserver_input_pipe ) )
{
    list_set(
        list,
        process_parse(
            input ) );
}
pclose( appaserver_input_pipe() );

/* Usage */
PROCESS *process_parse(
    char *input );

/* Process */
PROCESS *process_new(
    strdup( process_name ) );

/* Usage */
PROCESS *process_new(
    char *process_name );

/* Process */
PROCESS *process_calloc(
    void );

/* Usage */
PROCESS *process_seek(
    char *process_name,
    LIST *process_list );

/* Process */
for PROCESS *process in process_list
{
    if ( strcmp(
        process->process_name,
        process_name ) == 0 )
    {
        return process;
    }
}
return NULL;

/* Attributes */
char *process_name;
char *command_line;
char *notepad;
char *html_help_file_anchor;
int execution_count;
char *post_change_javascript;
char *process_set_display;
char *process_group;
char *preprompt_help_text;
char *javascript_filename;
JAVASCRIPT *javascript;
```



```
/* Usage */
boolean process_executable_okay(
    char *appaserver_parameter->mount_point,
    char *process->command_line );

/* Process */
if ( string_character_exists( command_line, '>' ) )
        return 0;
if ( string_character_exists( command_line, '<' ) )
        return 0;
if ( string_character_exists( command_line, ';' ) )
        return 0;
if ( string_character_exists( command_line, '&' ) )
        return 0;
if ( string_character_exists( command_line, '(' ) )
        return 0;
if ( string_character_exists( command_line, '`' ) )
        return 0;

column( command[], 0, command_line );

sprintf( system_string[], "which %s", command );

/* current_working_directory=$CGI_HOME */
void environment_prepend_path( "." );

char *which_string =
    string_pipe_fetch(
        system_string );

/* If not written yet, then okay */
if ( !which_string ) return 1;

/* Anything in $CGI_HOME is okay. */
char *directory =
    char *basename_directory(
        which_string );

if ( *directory == '.' ) return 1;

/* Can't reside in */
/* $APPASERVER_HOME/src_appaserver */
sprintf(
    src_directory[],
    "%s/src_appaserver",
    mount_point );

if ( string_strncmp(
        directory,
        src_directory ) == 0 )
{
    return 0;
}

/* Must reside in $APPASERVER_HOME/src_* */
sprintf(
    src_directory[],
    "%s/src_",
    mount_point );

if ( string_strncmp(
        directory,
        src_directory ) == 0 ) )
{
    return 1;
}

return 0;

/* Usage */
DICTIONARY *process_command_line_dictionary(
    DICTIONARY *drillthru_dictionary,
    DICTIONARY *prompt_dictionary );

/* Process */

/* Usage */
char *process_command_line_fetch(
    char *process_name );

/* Process */
PROCESS *process_fetch(
    process_name,
    /* Not fetching javascript */
    (char *)0 /* document_root */,
    (char *)0 /* relative_source_directory */,
    /* Not preparing to execute */
    0 /* not check_executable_inside_filesystem */,
    (char *)0 /* mount_point */ );

return process_fetch->command_line;

/* Usage */
LIST *process_role_process_name_list(
    char *role_primary_where );

/* Process */

/* Usage */
LIST *process_delimited_list(
    char *command_line );

/* Process */
LIST *list_fetch_pipe( command_line );

/* Usage */
void process_execution_count_increment(
    char *process_name );

/* Process */
void process_execution_increment_count( process_name );

/* Usage */
void process_increment_count(
    char *process_name );

/* Process */

/* Usage */
char *process_title_heading_tag(
    char *process_name );

/* Process */
```

```
/* Usage */
char *process_replace_command_line(
    const char *UPDATE_PREUPDATE_PREFIX,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *target_frame,
    char *state,
    char *process_name,
    char *many_folder_name,
    char *one_folder_name,
    char *related_column,
    char *update_results_string,
    char *update_error_string,
    DICTIONARY *operation_row_list_dictionary,
    DICTIONARY *dictionary_single_row,
    char *where_string,
    int row_number,
    int row_count,
    pid_t parent_process_id,
    LIST *primary_key_data_list,
    char *execute_yn,
    LIST *insert_datum_list,
    LIST *update_attribute_list,
    LIST *query_cell_list,
    char *appaserver_error_filespecification,
    char *input_command_line );

/* Process */
string_strcpy(
    command_line[],
    input_command_line,
    STRING_64K
        /* Synchronize with string_replace_command_line() */ );

void string_replace_command_line(
    command_line,
    application_name,
    PROCESS_APPLICATION_PLACEHOLDER );

void string_replace_command_line(
    command_line,
    session_key,
    PROCESS_SESSION_PLACEHOLDER );

void string_replace_command_line(
    command_line,
    login_name,
    PROCESS_LOGIN_NAME_PLACEHOLDER );

void string_replace_command_line(
    command_line,
    login_name,
    PROCESS_LOGIN_PLACEHOLDER );

void string_replace_command_line(
    command_line,
    role_name,
    PROCESS_ROLE_PLACEHOLDER );

void string_replace_command_line(
    command_line,
    folder_name,
    PROCESS_FOLDER_PLACEHOLDER );

void string_replace_command_line(
    command_line,
    folder_name,
    PROCESS_TABLE_PLACEHOLDER );

void string_replace_command_line(
    command_line,
    target_frame,
    PROCESS_TARGET_FRAME_PLACEHOLDER );

void string_replace_command_line(
    command_line,
    state,
    PROCESS_STATE_PLACEHOLDER );

void string_replace_command_line(
    command_line,
    many_folder_name,
    PROCESS_MANY_TABLE_PLACEHOLDER );

void string_replace_command_line(
    command_line,
    one_folder_name,
    PROCESS_ONE_TABLE_PLACEHOLDER );

void string_replace_command_line(
    command_line,
    related_column,
    PROCESS_RELATED_COLUMN_PLACEHOLDER );

void string_replace_command_line(
    command_line,
    update_results_string,
    PROCESS_UPDATE_RESULTS_PLACEHOLDER );

void string_replace_command_line(
    command_line,
    update_error_string,
    PROCESS_UPDATE_ERROR_PLACEHOLDER );

void process_replace_integer_command_line(
    command_line,
    row_number,
    PROCESS_ROW_NUMBER_PLACEHOLDER );

void process_replace_integer_command_line(
    command_line,
    row_count,
    PROCESS_ROW_COUNT_PLACEHOLDER );

/* Must preceed PROCESS_NAME_PLACEHOLDER */
void process_replace_integer_command_line(
    command_line,
    (int)parent_process_id,
    PROCESS_PID_PLACEHOLDER );

void string_replace_command_line(
    command_line,
    process_name,
    PROCESS_NAME_PLACEHOLDER );

void string_replace_command_line(
    command_line,
    list_display_delimited(
        primary_key_data_list,
        ATTRIBUTE_MULTI_KEY_DELIMITER ),
    PROCESS_PRIMARY_PLACEHOLDER );
```

```
/* process_replace_command_line() continued */
void process_replace_dictionary_command_line(
    command_line,
    operation_row_list_dictionary,
    DICTIONARY_ATTRIBUTE_DATUM_DELIMITER,
    DICTIONARY_ELEMENT_DELIMITER,
    PROCESS_DICTIONARY_PLACEHOLDER );

void dictionary_replace_command_line(
    command_line,
    dictionary_single_row );

void string_replace_command_line(
    command_line,
    where_string,
    PROCESS_WHERE_PLACEHOLDER );

void string_replace_command_line(
    command_line,
    execute_yn,
    APPASERVER_EXECUTE_YN );

for INSERT_DATUM *insert_datum in
    insert_datum_list
{
    string_replace_command_line(
        command_line,
        insert_datum->datum,
        insert_datum->attribute_name );
}

for UPDATE_ATTRIBUTE *update_attribute in
    update_attribute_list
{
    void string_with_space_search_replace(
        command_line,
        update_attribute->attribute_name,
        static char *string_double_quotes_around(
            update_attribute->post_datum ) );

    boolean update_changed_boolean(
        QUERY_IS_NULL,
        update_attribute->
            file_datum,
        update_attribute->
            post_datum );

    if ( update_changed_boolean() )
    {
        snprintf(
            preupdate_attribute_name[],
            sizeof ( preupdate_attribute_name ),
            "%s%s",
            UPDATE_PREUPDATE_PREFIX,
            update_attribute->attribute_name );

        string_with_space_search_replace(
            command_line,
            preupdate_attribute_name,
            static char *string_double_quotes_around(
                update_attribute->file_datum ) );
    }
}

void query_cell_command_line_replace(
    command_line /* in/out */,
    query_cell_list );

char *string_escape_dollar(
    command_line /* source */ );

if ( appaserver_error_filespecification )
{
    snprintf(
        command_line,
        sizeof ( command_line ),
        "%s 2>>%s",
        string_escape_dollar(),
            appaserver_error_filespecification );

    free( string_escape_dollar() );
    return strdup( command_line );
}
else
{
    return string_escape_dollar();
}

/* Usage */
void process_replace_dictionary_command_line(
    char *command_line /* in/out */,
    DICTIONARY *dictionary_single_row,
    const char *
        DICTIONARY_ATTRIBUTE_DATUM_DELIMITER,
    const char *
        DICTIONARY_ELEMENT_DELIMITER,
    const char *PROCESS_DICTIONARY_PLACEHOLDER );

/* Process */
char *dictionary_display_delimited(
    dictionary,
    DICTIONARY_ATTRIBUTE_DATUM_DELIMITER,
    DICTIONARY_ELEMENT_DELIMITER );

string_replace_command_line(
    command_line /* in/out */,
    dictionary_display_delimited() );

free( dictionary_display_delimited() );

/* Usage */
void process_replace_integer_command_line(
    char *command_line /* in/out */,
    int integer,
    const char *PROCESS_PLACEHOLDER );

/* Process */
```

JAVASCRIPT (1)

```
/* Usage */
JAVASCRIPT *javascript_new(
    char *javascript_filename,
    char *document_root,
    char *application_relative_source_directory() );

/* Process */
JAVASCRIPT *javascript_calloc( void );

char *javascript_script_tag(
    javascript_filename,
    document_root,
    application_relative_source_directory );

if ( !javascript_script_tag() ) exit( 1 );

/* Usage */
char *javascript_script_tag(
    char *javascript_filename,
    char *document_root,
    char *application_relative_source_directory );

/* Process */
char *javascript_home_source(
    javascript_filename,
    document_root );

boolean application_file_exists_boolean(
    javascript_home_source() );

if ( application_file_exists_boolean() )
{
    char *javascript_home_script_tag(
        javascript_filename );
}
else
{
    char *javascript_application_relative_script_tag(
        javascript_filename,
        document_root,
        application_relative_source_directory );
}

/* Usage */
char *javascript_home_source(
    char *javascript_filename,
    char *document_root );

/* Process */
fprintf( static home_source[],
    "%s/appaserver_home/javascript/",
    document_root,
    javascript_filename );

/* Usage */
char *javascript_home_script_tag(
    char *javascript_filename );

/* Process */
sprintf( script_tag[],
    "%s src=\"/appaserver_home/javascript/%s\"%s",
    javascript_open_tag(),
    javascript_filename,
    javascript_close_tag() );

/* Usage */
char *javascript_application_relative_script_tag(
    char *javascript_filename,
    char *document_root,
    char *application_relative_source_directory );

/* Process */
for(   int p = 0;
       piece(
          char relative_source_directory[],
          ':',
          application_relative_source_directory,
          p );
       p++ )
{
    char *javascript_relative_source(
        javascript_filename,
        document_root,
        relative_source_directory );

    boolean application_file_exists_boolean(
        javascript_relative_source() );

    if ( application_file_exists_boolean() )
    {
        char *javascript_relative_script_tag(
            javascript_filename,
            relative_source_directory );

        break;
    }
}

/* Usage */
char *javascript_relative_source(
    char *javascript_filename,
    char *document_root,
    char *relative_source_directory );

/* Process */
sprintf( static relative_source[],
    "%s/appaserver_home/%s/%s",
    document_root,
    relative_source_directory,
    javascript_filename );

/* Usage */
char *javascript_relative_script_tag(
    char *javascript_filename,
    char *relative_source_directory );

/* Process */
sprintf( script_tag[],
    "%s src=\"/appaserver_home/%s/%s\"%s",
    javascript_open_tag(),
    relative_source_directory,
    javascript_filename,
    javascript_close_tag() );

/* Attribute */
char *script_tag;
```

JAVASCRIPT (2)

```
/* Usage */
char *javascript_include_string(
    char *application_name,
    LIST *javascript_filename_list );

/* Process */
char string[];
char *ptr = string;

char *document_head_javascript_string();

ptr += sprintf( ptr,
    "%s",
    document_head_javascript_string() );

char *appaserver_parameter_document_root();

char *application_relative_source_directory(
    application_name );

for char *javascript_filename in
    javascript_filename_list
{
    JAVASCRIPT *javascript =
        javascript_new(
            javascript_filename,
            appaserver_parameter_document_root(),
            application_relative_source_directory() );
        ptr += sprintf( ptr,
            "\n%s",
            javascript->script_tag );
}

/* Usage */
char *javascript_replace(
    char *post_change_javascript,
    char *state,
    int row_number );

/* Process */
void javascript_replace_state(
    char *post_change_javascript /* in/out */,
    char *state );

void javascript_replace_row(
    post_change_javascript /* in/out */,
    row_number );

/* Usage */
char *javascript_post_change_append(
    char *original_post_change_javascript,
    char *post_change_javascript,
    boolean place_first );

/* Process */

/* Usage */
char *javascript_open_tag(
    void );

/* Process */
return
"<script language=JavaScript1.2";

/* Usage */
char *javascript_close_tag(
    void );

/* Process */
return
"></script>";

/* Usage */
void javascript_replace_row(
    char *post_change_javascript,
    int row_number );

/* Process */
char *string_itoa( row_number );

(void)string_search_replace(
    post_change_javascript,
    "$row",
    string_itoa() );
}
```

PROCESS_PARAMETER (1)

```
/*  Usage */
LIST *process_parameter_system_list(
    char *application_name,
    char *login_name,
    char *role_name,
    DICTIONARY *drillthru_dictionary,
    char *process_parameter_[set_]system_string() );

/* Process */
FILE *process_parameter_input_pipe(
    char *system_string );

for char *string_input(
    process_parameter_input_pipe() )
{
    list_set(
        list,
        process_parameter_parse(
            application_name,
            login_name,
            role_name,
            drillthru_dictionary,
            string_input() ) );
}

pclose( process_parameter_input_pipe() );

/* Usage */
PROCESS_PARAMETER *process_parameter_parse(
    char*application_name,
    char *login_name,
    char *role_name,
    DICTIONARY *drillthru_dictionary,
    char *input );

/* Process */
PROCESS_PARAMETER *process_parameter_calloc(
    void );

/* Note: all of the primary keys have at least a
    null string placeholder. */
if ( populate_drop_down_process_name )
{
    /* May just need the scalar variables */
    if ( application_name )
    {
        PROCESS_PARAMETER_DROP_DOWN *
            process_parameter_drop_down =
              process_parameter_drop_down_process_fetch(
                process_parameter->folder_name,
                process_parameter->prompt_name,
                process_parameter->drop_down_multi_select
                    /* multi_select_boolean */,
                process_parameter->
                    populate_drop_down_process_name );
    }
}
else
if (    strcmp( folder_name, "null" ) != 0
/* May just need the scalar variables */
&&   application_name
&&   login_name
&&   role_name )
{
    PROCESS_PARAMETER_DROP_DOWN *
        process_parameter_drop_down =
            process_parameter_drop_down_folder_fetch(
                application_name,
                login_name,
                role_name,
                process_parameter->folder_name,
                drillthru_dictionary,
                process_parameter->drop_down_multi_select
                    /* multi_select_boolean */ );
}
else
if ( strcmp( attribute_name, "null" ) != 0 )
{
    PROCESS_PARAMETER_ATTRIBUTE *
        process_parameter_attribute =
            process_parameter_attribute_fetch(
                process_parameter->attribute_name );
}
else
if (    strcmp( prompt_name, "null" ) != 0
&&   attribute_is_yes_no(
            prompt_name
                /* attribute_name */ ) )
{
    PROCESS_PARAMETER_YES_NO *
        process_parameter_yes_no =
            process_parameter_yes_no_fetch(
                process_parameter->prompt_name );
}
else
```

PROCESS_PARAMETER_DROP_DOWN

PROCESS_PARAMETER_ATTRIBUTE

PROCESS_PARAMETER_YES_NO

PROCESS_PARAMETER (2)

```
/* Process (continued) */
if ( strcmp( drop_down_prompt_name, "null" ) != 0 )
{
    PROCESS_PARAMETER_DROP_DOWN_PROMPT *
        process_parameter_drop_down_prompt =
            process_parameter_drop_down_prompt_fetch(
                process_parameter->
                    drop_down_prompt_name,
                process_parameter->
                    drop_down_multi_select
                    /* multi_select_boolean */ );
}
else
if (    strcmp( prompt_name, "null" ) != 0
&&  strcmp( drop_down_prompt_name, "null" ) == 0 )
{
    PROCESS_PARAMETER_PROMPT *
        process_parameter_prompt =
            process_parameter_prompt_fetch(
                process_parameter->prompt_name );
}

/* Attributes */
char *process_name;
char *folder_name;
char *attribute_name;
char *drop_down_prompt_name;
char *prompt_name;
int display_order;
boolean drop_down_multi_select;
boolean drillthru;
char *populate_drop_down_process_name;
char *populate_helper_process_name;

union {
PROCESS_PARAMETER_DROP_DOWN *
    process_parameter_drop_down;
PROCESS_PARAMETER_ATTRIBUTE *
    process_parameter_attribute;
PROCESS_PARAMETER_YES_NO *
    process_parameter_yes_no;
PROCESS_PARAMETER_DROP_DOWN_PROMPT *
    process_parameter_drop_down_prompt;
PROCESS_PARAMETER_PROMPT *
    process_parameter_prompt;
};
```

PROCESS_PARAMETER_DROP_DOWN_PROMPT

PROCESS_PARAMETER_PROMPT

PROCESS_PARAMETER (3)

```
/* Usage */
LIST *process_parameter_drop_down_prompt_list(
    LIST *process_parameter_list );

/* Process */

for PROCESS_PARAMETER *process_parameter in
    process_parameter_list()
{
    if ( strcmp(
            process_parameter->
                drop_down_prompt_name,
            "null" ) != 0 )
    {
        list_set(
            list,
            process_parameter->
                drop_down_prompt_name );
    }
}

/* Usage */
LIST *process_parameter_prompt_list(
    LIST *process_parameter_list );

/* Process */
for PROCESS_PARAMETER *process_parameter in
    process_parameter_list()
{
    if ( strcmp(
            process_parameter->prompt_name,
            "null" ) != 0 )
    {
        list_set(
            list,
            process_parameter->
                prompt_name );
    }
}

/* Usage */
LIST *process_parameter_folder_name_list(
    LIST *process_parameter_list );

/* Process */
for PROCESS_PARAMETER *process_parameter in
    process_parameter_list
{
    if ( strcmp(
            process_parameter->folder_name,
            "null" ) != 0 )
    {
        list_set(
            list,
            process_parameter->folder_name );
    }
}

/* Usage */
LIST *process_parameter_upload_filename_list(
    LIST *process_parameter_list );

/* Process */
for PROCESS_PARAMETER *process_parameter in
    process_parameter_list
{
    if (    process_parameter->
                process_parameter_prompt
        &&  process_parameter->
                process_parameter_prompt->
                upload_filename_boolean )
    {
        list_set(
            list,
            process_parameter->
                process_parameter_prompt->
                prompt_name );
    }
}

/* Usage */
LIST *process_parameter_date_name_list(
    LIST *process_parameter_list );

/* Process */
for PROCESS_PARAMETER *process_parameter in
    process_parameter_list
{
    if (    process_parameter->
                process_parameter_prompt
        &&  process_parameter->
                process_parameter_prompt->
                date_boolean )
    {
        list_set(
            list,
            process_parameter->
                process_parameter_prompt->
                prompt_name );
    }
}
```

PROCESS_PARAMETER (4)

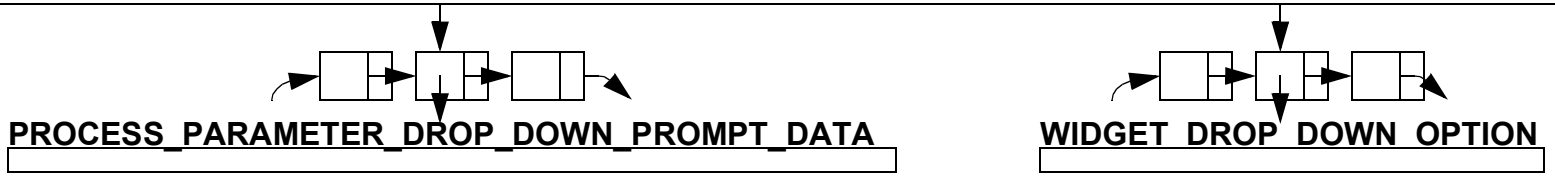
```
/* Usage */
boolean process_parameter_date_boolean(
    LIST *process_parameter_list );

/* Process */

/* Usage */
boolean process_parameter_drillthru_boolean(
    char *process_name );

/* Process */
char *process_parameter_where(
    process_name,
    1 /* is_drillthru */ );

char *system_string =
    process_parameter_system_string(
        "count(1)" /* select */,
        PROCESS_PARAMETER_TABLE,
        process_parameter_where() );

(boolean)atoi( string_pipe_fetch( system_string ) );
```

```
/* Usage */
LIST *process_parameter_list(
    char *process_name,
    boolean is_drillthru );

/* Process */
char *process_parameter_where(
    process_name,
    is_drillthru );

char *system_string =
    process_parameter_system_string(
        PROCESS_PARAMETER_SELECT,
        PROCESS_PARAMETER_TABLE,
        process_parmeter_where() );
}

LIST *process_parameter_system_list(
    (char *)0 /* application_name */,
    (char *)0 /* login_name */,
    (char *)0 /* role_name */,
    (DICTIONARY *)0 /* drillthru_dictionary */,
    system_string );
```

```
/* Usage */
char *process_parameter_system_string(
    const char *PROCESS_PARAMETER_SELECT,
    const char *PROCESS_PARAMETER_TABLE,
    char *process_parameter_where );

/* Process */

/* Usage */
static char *process_parameter_where(
    char *process_name,
    boolean is_drillthru );

/* Process */
snprintf(
        static parameter_where[],
        sizeof ( parameter_where ),
        "process = '%s' and "
        "ifnull(drillthru_yn,'n') = '%c'",
        process_name,
        (is_drillthru) ? 'y' : 'n' );
```

PROCESS_PARAMETER_DROP_DOWN_PROMPT

```
/* Usage */
PROCESS_PARAMETER_DROP_DOWN_PROMPT *
    process_parameter_drop_down_prompt_fetch(
        char *process_parameter->drop_down_prompt_name,
        boolean multi_select_boolean );

/* Process */
char *process_parameter_drop_down_prompt_primary_where(
    char *drop_down_prompt_name );

char *process_parameter_drop_down_prompt_system_string(
    char *PROCESS_PARAMETER_DROP_DOWN_PROMPT_SELECT,
    char *PROCESS_PARAMETER_DROP_DOWN_PROMPT_TABLE,
    char *process_parameter_drop_down_prompt_primary_where() );

char *string_pipe_input(
    process_parameter_drop_down_prompt_system_string() );

PROCESS_PARAMETER_DROP_DOWN_PROMPT *
    process_parameter_drop_down_prompt_parse(
        multi_select_boolean,
        drop_down_prompt_name,
        process_parameter_drop_down_prompt_primary_where(),
        string_pipe_input() );

/* Usage */
PROCESS_PARAMETER_DROP_DOWN_PROMPT *
    process_parameter_drop_down_prompt_parse(
        char *drop_down_prompt_name,
        boolean multi_select_boolean,
        process_parameter_drop_down_prompt_primary_where(),
        char *string_pipe_input() );

/* Process */
PROCESS_PARAMETER_DROP_DOWN_PROMPT *
    process_parameter_drop_down_prompt_calloc(
        void );

LIST *data_list =
    process_parameter_drop_down_prompt_data_list(
        drop_down_prompt_name,
        process_parameter_drop_down_prompt_primary_where() );

LIST *process_parameter_drop_down_prompt_data_option_list(
    data_list );

/* Attributes */
char *drop_down_prompt_name;
boolean multi_select_boolean;
char *hint_message;
char *optional_display;
LIST *data_list;
LIST *data_option_list;
```



PROCESS_PARAMETER_DROP_DOWN_PROMPT_DATA

WIDGET_DROP_DOWN_OPTION

PROCESS_PARAMETER_DROP_DOWN_PROMPT_DATA (1)

```
/* Usage */
LIST *process_parameter_drop_down_prompt_data_list(
    char *process_parameter->drop_down_prompt_name,
    char *process_parameter_drop_down_prompt_primary_where() );

/* Process */
char *process_parameter_drop_down_prompt_data_system_string(
    char *PROCESS_PARAMETER_DROP_DOWN_PROMPT_DATA_SELECT,
    char *PROCESS_PARAMETER_DROP_DOWN_PROMPT_TABLE,
    char *process_parameter_drop_down_prompt_primary_where() );

return
LIST *process_parameter_drop_down_prompt_data_system_list(
    drop_down_prompt_name,
    process_parameter_drop_down_prompt_system_string() );

/* Usage */
LIST *process_parameter_drop_down_prompt_data_system_list(
    char *drop_down_prompt_name,
    char *process_parameter_drop_down_prompt_system_string() );

/* Process */
FILE *process_parameter_drop_down_prompt_data_pipe(
    char *process_parameter_drop_down_prompt_system_string );

list = list_new();

for char *string_input( process_parameter_drop_down_prompt_data_pipe() )
{
    list_set(
        list,
        PROCESS_PARAMETER_DROP_DOWN_PROMPT_DATA *
            process_parameter_drop_down_prompt_data_parse(
                char *drop_down_prompt_name,
                char *string_input() ) );
}

pclose( process_parameter_drop_down_prompt_data_pipe() );

/* Usage */
PROCESS_PARAMETER_DROP_DOWN_PROMPT_DATA *
    process_parameter_drop_down_prompt_data_parse(
        char *drop_down_prompt_name,
        char *string_input() );

/* Process */
PROCESS_PARAMETER_DROP_DOWN_PROMPT_DATA *
    process_parameter_drop_down_prompt_data_new(
        drop_down_prompt_name,
        strdup( drop_down_prompt_data ) );

/* Usage */
PROCESS_PARAMETER_DROP_DOWN_PROMPT_DATA *
    process_parameter_drop_down_prompt_data_new(
        char *drop_down_prompt_name,
        char *strdup( drop_down_prompt_data ) );

/* Process */
PROCESS_PARAMETER_DROP_DOWN_PROMPT_DATA *
    process_parameter_drop_down_prompt_data_calloc(
        void );
```

PROCESS_PARAMETER_DROP_DOWN_PROMPT_DATA (2)

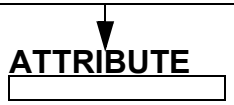
```
/* Usage */
LIST *process_parameter_drop_down_prompt_member_data_list(
    char *process_parameter_drop_down_prompt_set_member_name(),
    LIST *member_name_list );

/* Process */

/* Usage */
LIST *process_parameter_drop_down_prompt_data_in_clause_list(
    const char *PROCESS_PARAMETER_DROP_DOWN_PROMPT_DATA_TABLE,
    LIST *process_parameter_drop_down_prompt_list() );

/* Process */
snprintf(
    where[],
    sizeof ( where ),
    “drop_down_prompt_data in (%s)”,
    char *list_single_quotes_around_string(
        process_parameter_drop_down_prompt_list ) );

snprintf(
    system_string[],
    sizeof ( system_string ),
    “select.sh %s %s \”%s\””,
    “drop_down_prompt_data”,
    PROCESS_PARAMETER_DROP_DOWN_PROMPT_DATA_TABLE,
    where );

return LIST *list_pipe_fetch( system_string );

/* Usage */
LIST *process_parameter_drop_down_prompt_data_option_list(
    LIST *process_parameter_drop_down_prompt_data_list() );

/* Process */
for PROCESS_PARAMETER_DROP_DOWN_PROMPT_DATA *
    process_parameter_drop_down_prompt_data in
        process_parameter_drop_down_prompt_data_list
{
    WIDGET_DROP_DOWN_OPTION *widget_drop_down_option_new(
            process_parameter_drop_down_prompt_data->
                drop_down_prompt_data
                    /* string */ );

    list_set(
        option_list,
        widget_drop_down_option_new() );
}

/* Attributes */
char *drop_down_prompt_name;
char *drop_down_prompt_data;
int display_order;
```

PROCESS_PARAMETER_PROMPT

```
/* Usage */
PROCESS_PARAMETER_PROMPT *
    process_parameter_prompt_fetch(
        char *process_parameter->prompt_name );

/* Process */
char *process_parameter_prompt_primary_where(
    char *prompt_name );

char *process_parameter_prompt_system_string(
    char *PROCESS_PARAMETER_PROMPT_SELECT,
    char *PROCESS_PARAMETER_PROMPT_TABLE,
    char *process_parameter_prompt_primary_where() );

char *string_pipe_input(
    char *process_parameter_prompt_system_string() )

PROCESS_PARAMETER_PROMPT *
    process_parameter_prompt_parse(
        char *prompt_name,
        char *string_pipe_input() );

PROCESS_PARAMETER_PROMPT *
    process_parameter_prompt_new(
        char *prompt_name );

/* Usage */
PROCESS_PARAMETER_PROMPT *
    process_parameter_prompt_new(
        char *prompt_name );

/* Process */
PROCESS_PARAMETER_PROMPT *
    process_parameter_prompt_calloc(
        void );

/* Attributes */
char *prompt_name;
int input_width;
char *hint_message;
boolean upload_filename_boolean;
boolean date_boolean;
```

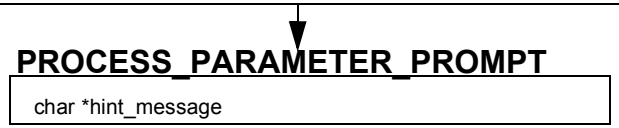
PROCESS_PARAMETER_ATTRIBUTE

```
/* Usage */
PROCESS_PARAMETER_ATTRIBUTE *
    process_parameter_attribute_fetch(
        char *process_parameter->attribute_name );

/* Process */
PROCESS_PARAMETER_ATTRIBUTE *
    process_parameter_attribute_calloc(
        void );

ATTRIBUTE *attribute =
    attribute_fetch(
        attribute_name );

/* Attribute */
ATTRIBUTE *attribute;
```



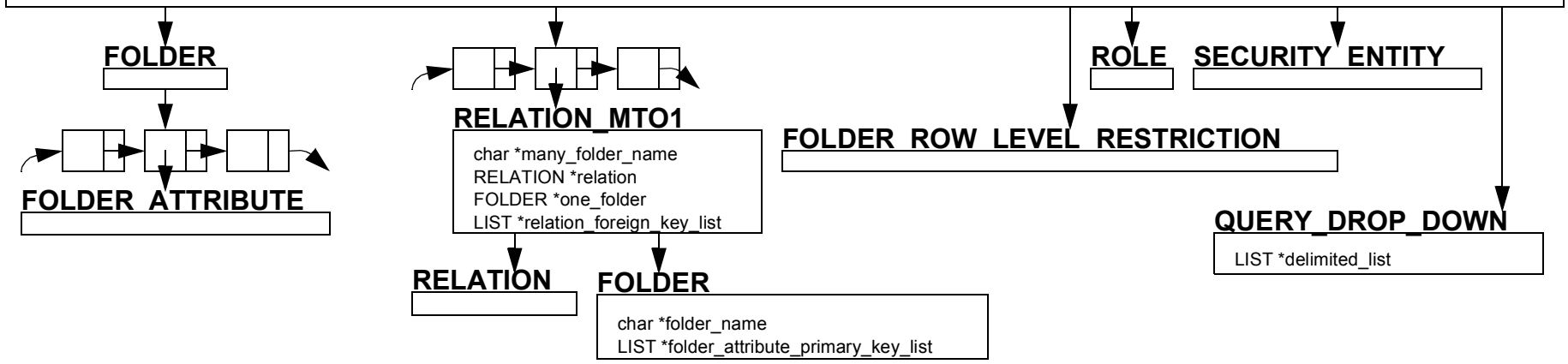
PROCESS_PARAMETER_YES_NO

```
/* Usage */
PROCESS_PARAMETER_YES_NO *
    process_parameter_yes_no_fetch(
        char *process_parameter->prompt_name );

/* Process */
PROCESS_PARAMETER_YES_NO *
    process_parameter_yes_no_calloc(
        void );

PROCESS_PARAMETER_PROMPT *
    process_parameter_prompt =
        process_parameter_prompt_fetch(
            prompt_name );

/* Attributes */
char *prompt_name;
PROCESS_PARAMETER_PROMPT *
    process_parameter_prompt;
```



PROCESS_PARAMETER_DROP_DOWN (1)

```
/* Usage */
PROCESS_PARAMETER_DROP_DOWN *
   process_parameter_drop_down_folder_fetch(
      char *application_name,
      char *login_name,
      char *role_name,
      char *folder_name,
      DICTIONARY *drillthru_dictionary,
      boolean multi_select_boolean );

/* Process */
PROCESS_PARAMETER_DROP_DOWN *
   process_parameter_drop_down_calloc();

FOLDER *folder =
   folder_fetch(
      folder_name,
      (LIST *)0 /*exclude_attribute_name_list */,
      /* Sets folder_attribute_primary_list */
      /* Sets folder_attribute_primary_key_list */
      /* Sets folder_attribute_name_list */
      1 /* fetch_folder_attribute_list */,
      1 /* fetch_attribute */,
      1 /* cache_boolean */ );

boolean process_parameter_drop_down_fetch_mto1(
   (boolean) dictionary_length( drillthru_dictionary )
      /* drillthru_dictionary_length */ );

if ( process_parameter_drop_down_fetch_mto1() )
{
      folder_name
         /* many_folder_name */,
      folder->folder_attribute_primary_key_list
         /* many_folder_primary_key_list */ );
}

char *process_parameter_drop_down_name(
   ATTRIBUTE_MULTI_KEY_DELIMITER,
   (char *)0 /* folder_name */,
   (char *)0 /* prompt_name */,
   folder->folder_attribute_primary_key_list );

FOLDER_ROW_LEVEL_RESTRICTION *
   folder_row_level_restriction =
      folder_row_level_restriction_fetch(
         folder_name );

ROLE *role =
   role_fetch(
      role_name,
      0 /* not fetch_role_attribute_exclude_list */ );

SECURITY_ENTITY *security_entity =
   security_entity_new(
      login_name,
      folder_name,
      folder_row_level_restriction->
         non_owner_forbid,
      role->override_row_restrictions );

QUERY_DROP_DOWN *query_drop_down_new(
   application_name,
   (char *)0 /* session_key */,
   login_name,
   (char *)0 /* role_name */,
   (char *)0 /* state */,
   (char *)0 /* many_folder_name */,
   folder_name /* one_folder_name */,
   (char *)0 /* related_column */,
   folder->folder_attribute_list
      /* one_folder_attribute_list */,
   folder->populate_drop_down_process_name,
   relation_mto1_list()
      /* relation_mto1_to_one */,
   drillthru_dictionary
      /* drop_down_dictionary */,
   security_entity,
   (LIST *)0 /* common_name_list */ );

LIST *widget_drop_down_option_list(
   ATTRIBUTE_MULTI_KEY_DELIMITER,
   WIDGET_DROP_DOWN_LABEL_DELIMITER,
   WIDGET_DROP_DOWN_EXTRA_DELIMITER,
   WIDGET_DROP_DOWN_DASH_DELIMITER,
   query_drop_down_new()->delimited_list,
   folder->no_initial_capital );
```

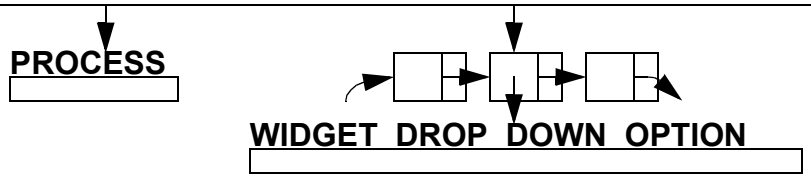


PROCESS_PARAMETER_DROP_DOWN (2)

```
/* Usage */
PROCESS_PARAMETER_DROP_DOWN *
   process_parameter_drop_down_process_fetch(
      char *process_parameter->folder_name,
      char *process_parameter->prompt_name,
      boolean process_parameter->
         multi_select_boolean,
      char *process_parameter->
         populate_drop_down_process_name );

/* Process */
PROCESS_PARAMETER_DROP_DOWN *
   process_parameter_drop_down_calloc( void );

char *process_parameter_drop_down_name(
   ATTRIBUTE_MULTI_KEY_DELIMITER,
   folder_name,
   prompt_name,
   (LIST *)0 /* folder_attribute_primary_key_list */ );

PROCESS *process =
   process_fetch(
      populate_drop_down_process_name,
      (char *)0 /* document_root */,
      (char *)0 /*relative_source_directory */,
      1 /* check_executable_inside_filesystem */,
      appaserver_parameter_mount_point() );
```

```
LIST *delimited_list =
   list_pipe(
      process->command_line );

boolean folder_no_initial_capital(
   folder_name );

LIST *widget_drop_down_option_list(
   ATTRIBUTE_MULTI_KEY_DELIMITER,
   WIDGET_DROP_DOWN_LABEL_DELIMITER,
   WIDGET_DROP_DOWN_EXTRA_DELIMITER,
   WIDGET_DROP_DOWN_DASH_DELIMITER,
   delimited_list,
   folder_no_initial_capital() );

/* Usage */
char *process_parameter_drop_down_name(
   const char ATTRIBUTE_MULTI_KEY_DELIMITER,
   char *folder_name,
   char *prompt_name,
   LIST *folder_attribute_primary_key_list );

/* Process */
if (   prompt_name
&&  strcmp( prompt_name, "null" ) != 0 )
{
   return prompt_name;
}
```

```
if (   folder_name
&&  strcmp( folder_name, "null" ) != 0 )
{
   LIST *folder_attribute_primary_key_list =
      folder_attribute_fetch_primary_key_list(
         folder_name );
}

char *list_display_delimited(
   folder_attribute_primary_key_list,
   ATTRIBUTE_MULTI_KEY_DELIMITER );

/* Attributes */
boolean multi_select_boolean;
FOLDER *folder;
boolean fetch_mto1;
LIST *relation_mto1_list;
char *name;
FOLDER_ROW_LEVEL_RESTRICTION *
   folder_row_level_restriction;
ROLE *role;
SECURITY_ENTITY *security_entity;
QUERY_DROP_DOWN *query_drop_down;
PROCESS *process;
LIST *delimited_list;
boolean folder_no_initial_capital;
LIST *widget_drop_down_option_list;
```



AGGREGATE_LEVEL

```
/* Usage */
enum aggregate_level aggregate_level_get(
    char *aggregate_level_string );

/* Process */

/* Usage */
enum aggregate_level
    aggregate_level_dictionary_extract(
        DICTIONARY *dictionary );

/* Process */
char *dictionary_get(
    AGGREGATE_LEVEL_LABEL,
    dictionary );

enum aggregate_level_get(
    dictionary_get() );

/* Usage */
char *aggregate_level_string(
    enum aggregate_level aggregate_level );

/* Process */

/* Usage */
char *aggregate_level_summation_heading(
    enum aggregate_level aggregate_level,
    char heading_delimiter );

/* Process */

/* Attributes */
```

aggregate_level_none	delta
real_time	daily
moving	weekly
half_hour	monthly
hourly	annually

AGGREGATE_STATISTIC

```
/* Usage */
enum aggregate_statistic aggregate_statistic_get(
    char *aggregate_statistic_string,
    enum aggregate_level aggregate_level);

/* Process */

/* Usage */
enum aggregate_statistic
    aggregate_statistic_dictionary_extract(
        DICTIONARY *dictionary,
        boolean aggregation_sum_boolean,
        enum aggregate_level aggregate_level );

/* Process */
char *dictionary_get(
    AGGREGATE_STATISTIC_LABEL,
    dictionary );
```

```
enum aggregate_statistic aggregate_statistic =
    aggregate_statistic_get(
        dictionary_get(),
        aggregate_level );

if (    aggregate_statistic == aggregate_statistic_none
&&  aggregate_level != aggregate_level_none
&&  aggregate_level != real_time )
{
    enum aggregate_statistic
        aggregate_statistic =
            aggregate_statistic_infer(
                aggregation_sum_boolean );
}

/* Usage */
enum aggregate_statistic
    aggregate_statistic_infer(
        boolean aggregation_sum_boolean );

/* Process */
if ( aggregation_sum_boolean )
    aggregate_statistic = sum;
else
    aggregate_statistic = average;

/* Usage */
char *aggregate_statistic_string(
    enum aggregate_statistic aggregate_statistic );

/* Process */

/* Attributes */
```

aggregate_statistic_none
average
sum
minimum
median
maximum
range
standard_deviation
non_zero_percent

OUTPUT_MEDIUM

```
/* Usage */
enum output_medium output_medium_get(
    char *output_medium_string );

/* Process */

/* Usage */
```

```
enum output_medium
    output_medium_dictionary_extract(
        DICTIONARY *dictionary );

/* Process */
char *dictionary_get(
    “output_medium”,
```

```
    dictionary );

enum output_medium_get(
    char *dictionary_get()
        /* output_medium_string */ );

/* Attributes */
```

output_medium_stdout
text_file
table
spreadsheet
gracechart
googlechart

ACCUMULATE_BOOLEAN

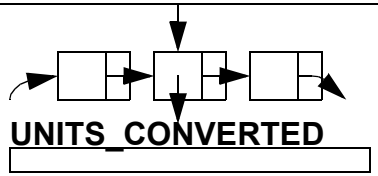
```
/* Usage */
boolean accumulate_boolean_get(
      char *accumulate_boolean_string );

/* Process */
return
( accumulate_boolean_string &&
  *accumulate_boolean_string == 'y' );
```

```
/* Usage */
boolean accumulate_boolean_dictionary_extract(
            DICTIONARY *dictionary );

/* Process */
char *dictionary_get(
      ACCUMULATE_BOOLEAN_LABEL,
      dictionary );
```

```
boolean accumulate_boolean_get(
      dictionary_get() );

/* Attributes */
```

UNITS

```
/* Usage */
LIST *units_list(
    boolean fetch_converted_list );

/* Process */
char *appaserver_system_string(
    “units” /* select */,
    UNITS_TABLE,
    (char *)0 /* where */ );

FILE *appaserver_input_pipe(
    appaserver_system_string() );

LIST *list = list_new();

for input[] in string_input( appaserver_input_pipe() )
{
    list_set(
        list,
        units_new(
            strdup( input ) /* units_name */,
            fetch_converted_list ) );
}

pclose( appaserver_input_pipe() );

/* Usage */
UNITS *units_new(
    char *units_name,
    boolean fetch_converted_list );

/* Process */
UNITS *units_calloc( void );

if ( fetch_converted_list )
{
    char *units_primary_where(
        char *units_name );

    LIST *units_converted_list(
        units_name,
        units_primary_where() );
}

/* Attributes */
char *units_name;
LIST *units_converted_list;
```



UNITS_ALIAS

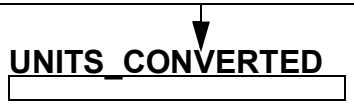
```
/* Usage */
LIST *units_alias_list( void );

/* Process */
char *appaserver_system_string(
    UNITS_ALIAS_SELECT,
    UNITS_ALIAS_TABLE,
    (char *)0 /* where */ );

FILE *appaserver_input_pipe(
    appaserver_system_string() );

LIST *list = list_new();

for input[] in string_input( appaserver_input_pipe() )
{
    UNITS_ALIAS *units_alias =
        units_alias_parse(
            input );
    if ( !units_alias )
    {
        pclose( appaserver_input_pipe(),
        exit( 1 );
    }

    list_set( list, units_alias );
}

pclose( appaserver_input_pipe() );

/* Usage */
UNITS_ALIAS *units_alias_parse(
    char *input );

/* Process */
UNITS_ALIAS *units_alias_calloc( void );

/* Usage */
UNITS_ALIAS *units_alias_seek(
    char *units_alias_name,
    LIST *units_alias_list() );

/* Process */
for UNITS_ALIAS *units_alias in units_alias_list
{
    if ( strcmp(
        units_alias_name,
        units_alias->units_alias_name ) == 0 )
    {
        return units_alias;
    }
}

return (UNITS_ALIAS *)0;

/* Attributes */
char *units_alias_name;
char *units_name;
```

UNITS_CONVERTED

```
/* Usage */
LIST *units_converted_list(
    char *units_name,
    char *units_primary_where() );

/* Process */
char *appaserver_system_string(
    UNITS_CONVERTED_SELECT,
    UNITS_CONVERTED_TABLE,
    units_primary_where );

FILE *appaserver_input_pipe(
    appaserver_system_string() );

LIST *list = list_new();

for input[] in string_input( appaserver_input_pipe() )
{
    list_set(
        list,
        units_converted_parse(
            units_name,
            input ) );
}

pclose( appaserver_input_pipe() );

/* Usage */
UNITS_CONVERTED *units_converted_fetch(
    char *units_name,
    char *units_converted_name );

/* Process */
char *units_converted_primary_where(
    char *units_name,
    char *units_converted_name );

char *appaserver_system_string(
    UNITS_CONVERTED_SELECT,
    UNITS_CONVERTED_TABLE,
    units_converted_primary_where() );

char *string_pipe_input(
    char *appaserver_system_string() );

UNITS_CONVERTED *units_converted_parse(
    units_name,
    string_pipe_input() );

/* Usage */
UNITS_CONVERTED *units_converted_parse(
    char *units_name,
    char *input );

/* Process */
UNITS_CONVERTED *units_converted_calloc( void );

/* Usage */
char *units_converted_dictionary_extract(
    DICTIONARY *dictionary );

/* Process */
char *dictionary_get(
    UNITS_CONVERTED_LABEL,
    dictionary );

/* Attributes */
char *units_name;
char *units_converted_name;
double multiply_by;
```

UNITS_CONVERTED_STDIN

```
/* Usage */
UNITS_CONVERTED_STDIN *
   units_converted_stdin_new(
      char *units_name,
      char *units_converted_name,
      int value_piece,
      char delimiter,
      int station_piece /* -1 for not applied */ );

/* Process */
UNITS_CONVERTED_STDIN *
   units_converted_stdin_calloc(
      void );

units_name =
   char *units_converted_stdin_constant_spelling(
      char *units_name );

units_converted_name =
   char *units_converted_stdin_constant_spelling(
      units_converted_name );

boolean units_converted_stdin_temperature_boolean(
   char *units_name );

boolean units_converted_stdin_navd88_boolean(
   char *UNITS_CONVERTED_STDIN_NAVD88_NAME,
   char *units_converted_name );

if (   units_converted_stdin_navd88_boolean()
&&  station_piece == -1 )
      exit( 1 );

if ( units_converted_stdin_navd88_boolean() )
{
   DICTIONARY *
      units_converted_stdin_station_navd88_dictionary(
         char *APPASERVER_STATION_TABLE,
         char *APPASERVER_NAVD88_ATTRIBUTE );
}

if (   !units_converted_stdin_temperature_boolean()
&&  !units_converted_stdin_navd88_boolean() )
{
   UNITS_CONVERTED *units_converted =
      units_converted_fetch(
         units_name,
         units_converted_name );
}

/* Driver */
void units_converted_stdin_execute(
   UNITS_CONVERTED_STDIN *
      units_converted_stdin );

/* Process */
UNITS_CONVERTED_ROW *
   units_converted_row =
      units_converted_row_calloc();

for input[] in string_input( stdin )
{
   units_converted_row =
      units_converted_row_parse(
         input /* in/out */,
         units_converted_row /* in/out */,
         units_converted_stdin );

   printf(
      "%s\n",
      units_converted_row->
         output_string );
}

/* Attributes */
char *units_name;
char *units_converted_name;
int value_piece;
char delimiter;
int station_piece;
boolean temperature_boolean;
boolean navd88_boolean;
DICTIONARY *station_navd88_dictionary;
UNITS_CONVERTED *units_converted;
```



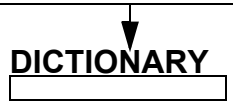
UNITS_CONVERTED_ROW

```
/* Usage */
UNITS_CONVERTED_ROW *
    units_converted_row_calloc(
        void );

/* Process */

/* Usage */
UNITS_CONVERTED_ROW *
    units_converted_row_parse(
        char *input /* in/out */
        UNITS_CONVERTED_ROW *units_converted_row
            /* in/out */,
        UNITS_CONVERTED_STDIN *
            units_converted_stdin );

/* Process */
char *units_converted_row_value_string(
    int units_converted_stdin->value_piece,
    char units_converted_stdin->delimiter,
    char *input );

if ( !units_converted_row_value_string() )
{
        char *output_string = input;
        return this;
}

double units_converted_row_old_value(
    char *units_converted_row_value_string() );

if (    units_converted_stdin->navd88_boolean
&&  units_converted_stdin->station_piece > -1 )
{
    char *units_converted_row_station_string(
        int units_converted_stdin->station_piece,
        char units_converted_stdin->delimiter,
        char *input );

    if ( !units_converted_row_station_string() )
            exit( 1 );

    double units_converted_row_navd88_new_value(
        units_converted_stdin->
            station_navd88_dictionary,
        units_converted_row_old_value(),
        units_converted_row_station_string() );

    char *units_converted_row_output_string(
        input /* in/out */,
        units_converted_stdin->value_piece,
        units_converted_stdin->delimiter,
        units_converted_row_navd88_new_value() );
}
else
if ( units_converted_stdin->temperature_boolean )
{
    double
        units_converted_row_temperature_new_value(
            char *units_converted_stdin->
                units_name,
            char *units_converted_stdin->
                units_converted_name,
            double units_converted_row_old_value() );

    char *units_converted_row_output_string(
        input /* in/out */,
        units_converted_stdin->value_piece,
        units_converted_stdin->delimiter,
        units_converted_row_temperature_new_value() );
}
else
if ( units_converted_stdin->units_converted )
{
    double units_converted_row_new_value(
        double units_converted_stdin->
            units_converted->
            multiply_by,
        double units_converted_row_old_value() );

    char *units_converted_row_output_string(
        input /* in/out */,
        units_converted_stdin->value_piece,
        units_converted_stdin->delimiter,
        units_converted_row_new_value() );
}
else
{
    exit(1 );
}

/* Usage */
double units_converted_row_navd88_new_value(
    DICTIONARY *units_converted_stdin->
        station_navd88_dictionary,
    double units_converted_row_old_value(),
    char *units_converted_row_station_string() );

/* Process */

/* Usage */
char *units_converted_row_output_string(
    char *input /* in/out */,
    int units_converted_stdin->value_piece,
    char units_converted_stdin->delimiter,
    double new_value );

/* Process */
char *string_ftoa(
    new_value,
    4 /* decimal_places */ );

piece_replace(
    input /* in/out */,
    delimiter,
    string_ftoa(),
    value_piece );

return input;

/* Attributes */
char *value_string;
double old_value;
char *station_string;
double new_value;
char *output_string;
```

DICTIONARY_SEPARATE (1)

```
/* Usage */
DICTIONARY_SEPARATE *
    dictionary_separate_string_new(
        DICTIONARY *post_dictionary->
            original_post_dictionary );

/* Process */
```

```
DICTIONARY_SEPARATE *dictionary_separate_calloc(
    void );

DICTIONARY *dictionary_separate_non_prefixed(
    DICTIONARY_SEPARATE_SORT_PREFIX,
    DICTIONARY_SEPARATE_QUERY_PREFIX,
    DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
    DICTIONARY_SEPARATE_IGNORE_PREFIX,
    DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
    DICTIONARY_SEPARATE_PAIR_PREFIX,
    original_post_dictionary );

/* Attribute */
DICTIONARY *non_prefixed_dictionary;
```



DICTIONARY_SEPARATE (2)

```
/* Usage */
DICTIONARY *dictionary_separate_row_number(
    DICTIONARY *destination_dictionary,
    LIST *folder_attribute_name_list,
    DICTIONARY *source_dictionary,
    int row_number );

/* Process */
for char *attribute_name in folder_attribute_name_list
{
    snprintf(
        key[],
        sizeof ( key ),
        "%s_%d",
        attribute_name,
        row_number );

    char *dictionary_get(
        key,
        source_dictionary );

    if ( dictionary_get() )
    {
        dictionary_set(
            destination_dictionary,
            source_dictionary->
                hash_table->
                last_cell_retrieved->
                key,
            dictionary_get() );
}

/* Usage */
DICTIONARY *dictionary_separate_prompt(
    const char *PROMPT_LOOKUP_FROM_PREFIX,
    const char *PROMPT_LOOKUP_TO_PREFIX,
    LIST *relation_mto1_list,
    LIST *folder_attribute_name_list,
    DICTIONARY *dictionary );

/* Process */
DICTIONARY *dictionary_small();

for RELATION_MTO1 *relation_mto1
    in relation_mto1_list
{
    char *key =
        char *list_display_delimited(
            relation_mto1->
                relation_foreign_key_list,
            ATTRIBUTE_MULTI_KEY_DELIMITER );

    char *dictionary_get( key, dictionary );

    if ( dictionary_get() )
    {
        dictionary_set(
            dictionary_small(),
            key,
            dictionary_get() );
    }
    else
    {
        free( key );
    }
}

for char *attribute_name in folder_attribute_name_list
{
    char *dictionary_get(
        attribute_name,
        dictionary );

    if ( dictionary_get() )
    {
        dictionary_set(
            dictionary_small(),
            attribute_name,
            dictionary_get() );

        continue;
    }

    char *dictionary_separate_from_to_key(
        PROMPT_LOOKUP_FROM_PREFIX,
        attribute_name );

    char *dictionary_get(
        dictionary_separate_from_to_key(),
        dictionary );

    if ( dictionary_get() )
    {
        char *strdup( dictionary_separate_from_to_key() );

        dictionary_set(
            dictionary_small(),
            strdup(),
            dictionary_get() );

        dictionary_set(
            dictionary_small(),
            attribute_name,
            dictionary_get() );
    }

    char *dictionary_separate_from_to_key(
        PROMPT_LOOKUP_TO_PREFIX,
        attribute_name );

    char *dictionary_get(
        dictionary_separate_from_to_key(),
        dictionary );

    if ( dictionary_get() )
    {
        char *strdup( dictionary_separate_from_to_key() );

        dictionary_set(
            dictionary_small(),
            strdup(),
            dictionary_get() );
    }
}

/* Usage */
char *dictionary_separate_from_to_key(
    const char *prefix,
    char *attribute_name );

/* Process */
snprintf(
    static key[],
    sizeof ( key ),
    "%s%s",
    prefix,
    attribute_name );
```

DICTIONARY_SEPARATE (3)

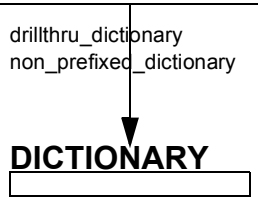
```
/* Usage */
DICTIONARY *dictionary_separate_multi_row(
    LIST *folder_attribute_name_list,
    DICTIONARY *dictionary );

/* Process */
int dictionary_highest_index( dictionary );

if ( dictionary_highest_index() < 1 ) return NULL;

DICTIONARY *destination_dictionary =
   dictionary_large();

for(   int row_number = 1;
       row_number <= dictionary_highest_index();
       row_number++ )
{
   DICTIONARY *dictionary_separate_row_number(
       destination_dictionary,
       folder_attribute_name_list,
       dictionary /* source_dictionary */,
       row_number );
}

/* Usage */
LIST *dictionary_separate_no_display_name_list(
   DICTIONARY *no_display_dictionary );

/* Process */
LIST *dictionary_key_list( no_display_dictionary );
```

DICTIONARY_SEPARATE (3)

```
/* Usage */
char *dictionary_separate_send_string(
    DICTIONARY *
        dictionary_separate_send_dictionary() );

/* Process */
char *dictionary_display_delimited(
    dictionary_separate_send_dictionary,
    DICTIONARY_ATTRIBUTE_DATUM_DELIMITER,
    DICTIONARY_ELEMENT_DELIMITER );

string_escape_character(
    escaped_dictionary_string[],
    dictionary_display_delimited(),
    '$' );

free( dictionary_display_delimited() );

/* Usage */
DICTIONARY *dictionary_separate_remove_prefix(
    char *prefix,
    DICTIONARY *dictionary );

/* Process */

/* Usage */
DICTIONARY *dictionary_separate_non_prefixed(
    const char *
        DICTIONARY_SEPARATE_SORT_PREFIX,
    const char *
        DICTIONARY_SEPARATE_QUERY_PREFIX,
    const char *
        DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
    const char *
        DICTIONARY_SEPARATE_IGNORE_PREFIX,
    const char *
        DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
    const char *
        DICTIONARY_SEPARATE_PAIR_PREFIX,
    DICTIONARY *dictionary );

/* Process */
LIST *dictionary_key_list();

DICTIONARY *dictionary_separate_parse_non_prefixed(
    DICTIONARY_SEPARATE_SORT_PREFIX,
    DICTIONARY_SEPARATE_QUERY_PREFIX,
    DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
    DICTIONARY_SEPARATE_IGNORE_PREFIX,
    DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
    DICTIONARY_SEPARATE_PAIR_PREFIX,
    dictionary,
    dictionary_key_list );

/* Usage */
DICTIONARY *dictionary_separate_operation(
    LIST *folder_operation_name_list,
    DICTIONARY *dictionary );

/* Process */

/* Usage */
char *dictionary_separate_hidden_html(
    char *prefix,
    DICTIONARY *dictionary );

/* Process */

/* Usage */
DICTIONARY *dictionary_separate_send_dictionary(
    DICTIONARY *sort_dictionary,
    const char *
        DICTIONARY_SEPARATE_SORT_PREFIX,
    DICTIONARY *query_dictionary,
    const char *
        DICTIONARY_SEPARATE_QUERY_PREFIX,
    DICTIONARY *drillthru_dictionary,
    const char *
        DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
    DICTIONARY *ignore_dictionary,
    const char *
        DICTIONARY_SEPARATE_IGNORE_PREFIX,
    DICTIONARY *no_display_dictionary,
    const char *
        DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
    DICTIONARY *pair_dictionary,
    const char *
        DICTIONARY_SEPARATE_PAIR_PREFIX,
    DICTIONARY *non_prefixed_dictionary );

/* Process */

/* Usage */
char *dictionary_separate_relation_key(
    char *QUERY_RELATION_OPERATOR_PREFIX,
    char *single_key );

/* Process */
```

DICTIONARY_SEPARATE_DRILLTHRU

```
/* Usage */
DICTIONARY_SEPARATE_DRILLTHRU *
    dictionary_separate_drillthru_new(
        DICTIONARY *post_dictionary->
            original_post_dictionary,
        char *application_name,
        char *login_name,
        LIST *folder_attribute_date_name_list(),
        LIST *folder_attribute_list );

/* Process */
DICTIONARY_SEPARATE_DRILLTHRU *
    dictionary_separate_drillthru_calloc(
        void );

if ( !dictionary_length( original_post_dictionary ) )
{
    return this;
}

DICTIONARY_SEPARATE_TRIM_DOUBLE_BRACKET *
    dictionary_separate_trim_double_bracket_new(
        original_post_dictionary /* in/out */ );

DICTIONARY_SEPARATE_PARSE *
    dictionary_separate_parse_new(
        DICTIONARY_SEPARATE_SORT_PREFIX,
        DICTIONARY_SEPARATE_QUERY_PREFIX,
        DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
        DICTIONARY_SEPARATE_IGNORE_PREFIX,
        DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
        DICTIONARY_SEPARATE_PAIR_PREFIX,
        ATTRIBUTE_MULTI_KEY_DELIMITER,
        dictionary_separate_trim_double_bracket_new()->
            dictionary );

DICTIONARY_SEPARATE_DATE_CONVERT *
    dictionary_separate_date_convert_prompt_new(
        PROMPT_LOOKUP_FROM_PREFIX,
        PROMPT_LOOKUP_TO_PREFIX,
        application_name,
        login_name,
        folder_attribute_date_name_list,
        dictionary_separate_parse_new() );

DICTIONARY_SEPARATE_SQL_INJECTION_ESCAPE *
    dictionary_separate_sql_injection_escape_new(
        folder_attribute_list,
        dictionary_separate_date_convert_prompt_new() );

DICTIONARY *drillthru_dictionary =
    dictionary_separate_sql_injection_escape_new()->
        drillthru_dictionary;

DICTIONARY *non_prefixed_dictionary =
    dictionary_separate_sql_injection_escape_new()->
        non_prefixed_dictionary;

/* Attributes */
DICTIONARY *drillthru_dictionary;
DICTIONARY *non_prefixed_dictionary;
```



DICTIONARY_SEPARATE_PROMPT_LOOKUP

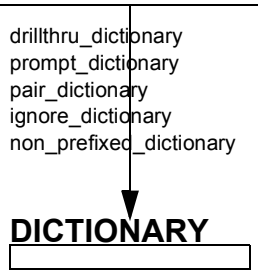
```
/* Usage */
DICTIONARY_SEPARATE_PROMPT_LOOKUP *
   dictionary_separate_prompt_lookup_new(
      DICTIONARY *post_dictionary->
         original_post_dictionary,
      char *application_name,
      char *login_name,
      LIST *relation_mto1_list,
      LIST *folder_attribute_name_list,
      LIST *folder_attribute_date_name_list(),
      LIST *folder_attribute_list );

/* Process */
DICTIONARY_SEPARATE_PROMPT_LOOKUP *
   dictionary_separate_prompt_calloc(
      void );

if ( !dictionary_length( original_post_dictionary ) )
{
   return this;
}

DICTIONARY_SEPARATE_TRIM_DOUBLE_BRACKET *
   dictionary_separate_trim_double_bracket_new(
      original_post_dictionary /* in/out */ );

DICTIONARY_SEPARATE_PARSE *
   dictionary_separate_parse_new(
      DICTIONARY_SEPARATE_SORT_PREFIX,
      DICTIONARY_SEPARATE_QUERY_PREFIX,
      DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
      DICTIONARY_SEPARATE_IGNORE_PREFIX,
      DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
      DICTIONARY_SEPARATE_PAIR_PREFIX,
      ATTRIBUTE_MULTI_KEY_DELIMITER,
      dictionary_separate_trim_double_bracket_new()->
         dictionary );

DICTIONARY_SEPARATE_DATE_CONVERT *
   dictionary_separate_date_convert_prompt_new(
      PROMPT_LOOKUP_FROM_PREFIX,
      PROMPT_LOOKUP_TO_PREFIX,
      application_name,
      login_name,
      folder_attribute_date_name_list,
      dictionary_separate_parse_new() );

DICTIONARY_SEPARATE_SQL_INJECTION_ESCAPE *
   dictionary_separate_sql_injection_escape_new(
      folder_attribute_list,
      dictionary_separate_date_convert_prompt_new() );

DICTIONARY *prompt_dictionary =
   dictionary_separate_prompt(
      PROMPT_LOOKUP_FROM_PREFIX,
      PROMPT_LOOKUP_TO_PREFIX,
      relation_mto1_list,
      folder_attribute_name_list,
      dictionary_separate_sql_injection_escape_new()->
         non_prefixed_dictionary );

DICTIONARY *non_prefixed_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      non_prefixed_dictionary );

/* Attributes */
DICTIONARY *prompt_dictionary;
DICTIONARY *non_prefixed_dictionary;
```

prompt_dictionary
non_prefixed_dictionary

DICTIONARY

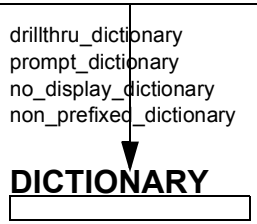
DICTIONARY_SEPARATE_POST_PROMPT_INSERT

```
/* Usage */
DICTIONARY_SEPARATE_POST_PROMPT_INSERT *
   dictionary_separate_post_prompt_insert_new(
      DICTIONARY *post_dictionary->
         original_post_dictionary,
      char *application_name,
      char *login_name,
      LIST *relation_mto1_list,
      LIST *folder_attribute_append_isa_list,
      LIST *folder_attribute_name_list,
      LIST *folder_attribute_date_name_list );

/* Process */
DICTIONARY_SEPARATE_POST_PROMPT_INSERT *
   dictionary_separate_post_prompt_insert_calloc(
      void );

if ( !dictionary_length( original_post_dictionary ) )
{
   return this;
}

DICTIONARY_SEPARATE_TRIM_DOUBLE_BRACKET *
   dictionary_separate_trim_double_bracket_new(
      original_post_dictionary /* in/out */ );

DICTIONARY_SEPARATE_PARSE *
   dictionary_separate_parse_new(
      DICTIONARY_SEPARATE_SORT_PREFIX,
      DICTIONARY_SEPARATE_QUERY_PREFIX,
      DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
      DICTIONARY_SEPARATE_IGNORE_PREFIX,
      DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
      DICTIONARY_SEPARATE_PAIR_PREFIX,
      ATTRIBUTE_MULTI_KEY_DELIMITER,
      dictionary_separate_trim_double_bracket_new()->
         dictionary );

DICTIONARY_SEPARATE_DATE_CONVERT *
   dictionary_separate_date_convert_prompt_new(
      PROMPT_LOOKUP_FROM_PREFIX,
      PROMPT_LOOKUP_TO_PREFIX,
      application_name,
      login_name,
      folder_attribute_date_name_list,
      dictionary_separate_parse_new() );

DICTIONARY_SEPARATE_SQL_INJECTION_ESCAPE *
   dictionary_separate_sql_injection_escape_new(
      folder_attribute_append_isa_list,
      dictionary_separate_date_convert_prompt_new() );

DICTIONARY *drillthru_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      drillthru_dictionary;

DICTIONARY *prompt_dictionary =
   dictionary_separate_prompt(
      PROMPT_LOOKUP_FROM_PREFIX,
      PROMPT_LOOKUP_TO_PREFIX,
      relation_mto1_list,
      folder_attribute_name_list,
      dictionary_separate_sql_injection_escape_new()->
         non_prefixed_dictionary );

DICTIONARY *pair_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      pair_dictionary;

DICTIONARY *ignore_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      ignore_dictionary;

DICTIONARY *non_prefixed_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      non_prefixed_dictionary;

LIST *ignore_name_list =
   dictionary_separate_no_display_name_list(
      ignore_dictionary /* no_display_dictionary */ );

/* Attributes */
DICTIONARY *drillthru_dictionary;
DICTIONARY *prompt_dictionary;
DICTIONARY *pair_dictionary;
DICTIONARY *ignore_dictionary;
DICTIONARY *non_prefixed_dictionary;
```



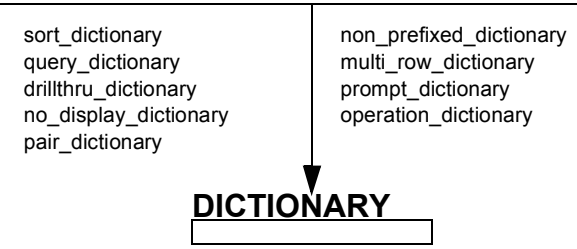
DICTIONARY_SEPARATE_POST_PROMPT_LOOKUP

```
/* Usage */
DICTIONARY_SEPARATE_POST_PROMPT_LOOKUP *
   dictionary_separate_post_prompt_lookup_new(
      DICTIONARY *post_dictionary->
         original_post_dictionary,
      char *application_name,
      char *login_name,
      LIST *relation_mto1_list(),
      LIST *folder_attribute_name_list(),
      LIST *folder_attribute_date_name_list(),
      LIST *folder_attribute_append_isa_list );

/* Process */
DICTIONARY_SEPARATE_POST_PROMPT_LOOKUP *
   dictionary_separate_post_prompt_lookup_calloc(
      void );

if ( !dictionary_length( original_post_dictionary ) )
{
   return this;
}

DICTIONARY_SEPARATE_TRIM_DOUBLE_BRACKET *
   dictionary_separate_trim_double_bracket_new(
      original_post_dictionary /* in/out */ );

DICTIONARY_SEPARATE_PARSE *
   dictionary_separate_parse_new(
      DICTIONARY_SEPARATE_SORT_PREFIX,
      DICTIONARY_SEPARATE_QUERY_PREFIX,
      DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
      DICTIONARY_SEPARATE_IGNORE_PREFIX,
      DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
      DICTIONARY_SEPARATE_PAIR_PREFIX,
      ATTRIBUTE_MULTI_KEY_DELIMITER,
      dictionary_separate_trim_double_bracket_new()->
         dictionary );

DICTIONARY_SEPARATE_DATE_CONVERT *
   dictionary_separate_date_convert_prompt_new(
      PROMPT_LOOKUP_FROM_PREFIX,
      PROMPT_LOOKUP_TO_PREFIX,
      application_name,
      login_name,
      folder_attribute_date_name_list,
      dictionary_separate_parse_new() );

DICTIONARY_SEPARATE_SQL_INJECTION_ESCAPE *
   dictionary_separate_sql_injection_escape_new(
      folder_attribute_append_isa_list,
      dictionary_separate_date_convert_prompt_new() );

DICTIONARY *drillthru_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      drillthru_dictionary;

DICTIONARY *no_display_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      no_display_dictionary;

DICTIONARY *non_prefixed_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      non_prefixed_dictionary;

DICTIONARY *prompt_dictionary =
   dictionary_separate_prompt(
      PROMPT_LOOKUP_FROM_PREFIX,
      PROMPT_LOOKUP_TO_PREFIX,
      relation_mto1_list,
      folder_attribute_name_list,
      non_prefixed_dictionary );

LIST *no_display_name_list =
   dictionary_separate_no_display_name_list(
      no_display_dictionary );

/* Attributes */
DICTIONARY *drillthru_dictionary;
DICTIONARY *no_display_dictionary;
DICTIONARY *non_prefixed_dictionary;
DICTIONARY *prompt_dictionary;
LIST *no_display_name_list;
```



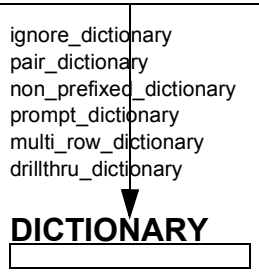
DICTIONARY_SEPARATE_POST_TABLE_EDIT

```
/* Usage */
DICTIONARY_SEPARATE_POST_TABLE_EDIT *
   dictionary_separate_post_table_edit_new(
      DICTIONARY *post_dictionary->
         original_post_dictionary,
      char *application_name,
      char *login_name,
      LIST *relation_mto1_list,
      LIST *folder_attribute_name_list,
      LIST *folder_operation_name_list,
      LIST *folder_attribute_date_name_list,
      LIST *folder_attribute_append_isa_list );

/* Process */
DICTIONARY_SEPARATE_POST_TABLE_EDIT *
   dictionary_separate_post_table_edit_calloc(
      void );

if ( !dictionary_length( original_post_dictionary ) )
{
   return this;
}

DICTIONARY_SEPARATE_TRIM_DOUBLE_BRACKET *
   dictionary_separate_trim_double_bracket_new(
      original_post_dictionary /* in/out */ );

DICTIONARY_SEPARATE_PARSE *
   dictionary_separate_parse_new(
      DICTIONARY_SEPARATE_SORT_PREFIX,
      DICTIONARY_SEPARATE_QUERY_PREFIX,
      DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
      DICTIONARY_SEPARATE_IGNORE_PREFIX,
      DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
      DICTIONARY_SEPARATE_PAIR_PREFIX,
      ATTRIBUTE_MULTI_KEY_DELIMITER,
      dictionary_separate_trim_double_bracket_new()->
         dictionary );

DICTIONARY_SEPARATE_DATE_CONVERT *
   dictionary_separate_date_convert_edit_new(
      application_name,
      login_name,
      folder_attribute_date_name_list,
      dictionary_separate_parse_new() );

DICTIONARY_SEPARATE_SQL_INJECTION_ESCAPE *
   dictionary_separate_sql_injection_escape_new(
      folder_attribute_append_isa_list,
      dictionary_separate_date_convert_edit_new() );

DICTIONARY *sort_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      sort_dictionary;

DICTIONARY *query_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      query_dictionary;

DICTIONARY *drillthru_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      drillthru_dictionary;

DICTIONARY *no_display_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      no_display_dictionary;

DICTIONARY *pair_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      pair_dictionary;

DICTIONARY *non_prefixed_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      non_prefixed_dictionary;

DICTIONARY *multi_row_dictionary =
   dictionary_separate_multi_row(
      folder_attribute_name_list,
      dictionary_separate_sql_injection_escape_new()->
         non_prefixed_dictionary );

DICTIONARY *prompt_dictionary =
   dictionary_separate_prompt(
      PROMPT_LOOKUP_FROM_PREFIX,
      PROMPT_LOOKUP_TO_PREFIX,
      relation_mto1_list,
      folder_attribute_name_list,
      dictionary_separate_sql_injection_escape_new()->
         non_prefixed_dictionary );

DICTIONARY *operation_dictionary =
   dictionary_separate_operation(
      folder_operation_name_list,
      dictionary_separate_sql_injection_escape_new()->
         non_prefixed_dictionary );

LIST *dictionary_separate_no_display_name_list(
   no_display_dictionary );

/* Attributes */
DICTIONARY *sort_dictionary;
DICTIONARY *query_dictionary;
DICTIONARY *drillthru_dictionary;
DICTIONARY *no_display_dictionary;
DICTIONARY *pair_dictionary;
DICTIONARY *non_prefixed_dictionary;
DICTIONARY *multi_row_dictionary;
DICTIONARY *prompt_dictionary;
DICTIONARY *operation_dictionary;
LIST *no_display_name_list;
```



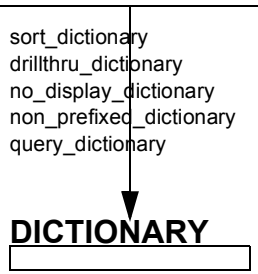
DICTIONARY_SEPARATE_POST_TABLE_INSERT

```
/* Usage */
DICTIONARY_SEPARATE_POST_TABLE_INSERT *
   dictionary_separate_post_table_insert_new(
      DICTIONARY *post_dictionary->
         original_post_dictionary,
      char *application_name,
      char *login_name,
      LIST *relation_mto1_list(),
      LIST *folder_attribute_name_list(),
      LIST *folder_attribute_date_name_list()
      LIST *folder_attribute_append_isa_list() );

/* Process */
DICTIONARY_SEPARATE_POST_TABLE_INSERT *
   dictionary_separate_post_table_insert_calloc(
      void );

if ( !dictionary_length( original_post_dictionary ) )
{
   return this;
}

DICTIONARY_SEPARATE_TRIM_DOUBLE_BRACKET *
   dictionary_separate_trim_double_bracket_new(
      original_post_dictionary );

DICTIONARY_SEPARATE_PARSE *
   dictionary_separate_parse_new(
      DICTIONARY_SEPARATE_SORT_PREFIX,
      DICTIONARY_SEPARATE_QUERY_PREFIX,
      DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
      DICTIONARY_SEPARATE_IGNORE_PREFIX,
      DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
      DICTIONARY_SEPARATE_PAIR_PREFIX,
      ATTRIBUTE_MULTI_KEY_DELIMITER,
      dictionary_separate_trim_double_bracket_new()->
         dictionary );

DICTIONARY_SEPARATE_DATE_CONVERT *
   dictionary_separate_date_convert_edit_new(
      application_name,
      login_name,
      folder_attribute_date_name_list,
      dictionary_separate_parse_new() );

DICTIONARY_SEPARATE_SQL_INJECTION_ESCAPE *
   dictionary_separate_sql_injection_escape_new(
      folder_attribute_append_isa_list,
      dictionary_separate_date_convert_edit_new() );

DICTIONARY *ignore_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      ignore_dictionary;

DICTIONARY *pair_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      pair_dictionary;

DICTIONARY *non_prefixed_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      non_prefixed_dictionary;

DICTIONARY *prompt_dictionary =
   dictionary_separate_prompt(
      PROMPT_LOOKUP_FROM_PREFIX,
      PROMPT_LOOKUP_TO_PREFIX,
      relation_mto1_list,
      folder_attribute_name_list,
      dictionary_separate_sql_injection_escape_new()->
         non_prefixed_dictionary );

DICTIONARY *multi_row_dictionary =
   dictionary_separate_multi_row(
      folder_attribute_name_list,
      dictionary_separate_sql_injection_escape_new()->
         non_prefixed_dictionary );

DICTIONARY *drillthru_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      drillthru_dictionary;

LIST *ignore_name_list =
   dictionary_separate_no_display_name_list(
      ignore_dictionary /* no_display_dictionary */ );

/* Attributes */
DICTIONARY *ignore_dictionary;
DICTIONARY *pair_dictionary;
DICTIONARY *non_prefixed_dictionary;
DICTIONARY *prompt_dictionary;
DICTIONARY *multi_row_dictionary;
DICTIONARY *drillthru_dictionary;
LIST *ignore_name_list;
```



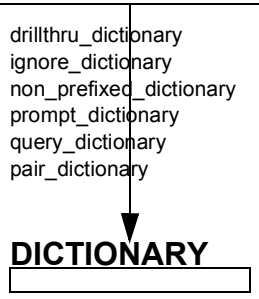
DICTIONARY_SEPARATE_TABLE_EDIT

```
/* Usage */
DICTIONARY_SEPARATE_TABLE_EDIT *
   dictionary_separate_table_edit_new(
      DICTIONARY *post_dictionary->
         original_post_dictionary /* in/out */,
      char *application_name,
      char *login_name,
      LIST *folder_attribute_date_name_list(),
      LIST *folder_attribute_append_isa_list );

/* Process */
DICTIONARY_SEPARATE_TABLE_EDIT *
   dictionary_separate_table_edit_calloc(
      void );

if ( !dictionary_length( original_post_dictionary ) )
{
   return this;
}

DICTIONARY_SEPARATE_TRIM_DOUBLE_BRACKET *
   dictionary_separate_trim_double_bracket_new(
      original_post_dictionary /* in/out */ );

DICTIONARY_SEPARATE_PARSE *
   dictionary_separate_parse_new(
      DICTIONARY_SEPARATE_SORT_PREFIX,
      DICTIONARY_SEPARATE_QUERY_PREFIX,
      DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
      DICTIONARY_SEPARATE_IGNORE_PREFIX,
      DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
      DICTIONARY_SEPARATE_PAIR_PREFIX,
      ATTRIBUTE_MULTI_KEY_DELIMITER,
      dictionary_separate_trim_double_bracket_new()->
         dictionary );

DICTIONARY_SEPARATE_DATE_CONVERT *
   dictionary_separate_date_convert_edit_new(
      application_name,
      login_name,
      folder_attribute_date_name_list,
      dictionary_separate_parse_new() );

DICTIONARY_SEPARATE_SQL_INJECTION_ESCAPE *
   dictionary_separate_sql_injection_escape_new(
      folder_attribute_append_isa_list,
      dictionary_separate_date_convert_edit_new() );

DICTIONARY *sort_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      sort_dictionary;

DICTIONARY *drillthru_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      drillthru_dictionary;

DICTIONARY *no_display_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      no_display_dictionary;

LIST *dictionary_separate_no_display_name_list(
   no_display_dictionary );

DICTIONARY *non_prefixed_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      non_prefixed_dictionary;

DICTIONARY *query_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      query_dictionary;

dictionary_append_dictionary(
   query_dictionary /* in/out */,
   non_prefixed_dictionary );

/* Attributes */
DICTIONARY *sort_dictionary;
DICTIONARY *drillthru_dictionary;
DICTIONARY *no_display_dictionary;
LIST *no_display_name_list;
DICTIONARY *non_prefixed_dictionary;
DICTIONARY *query_dictionary;
```



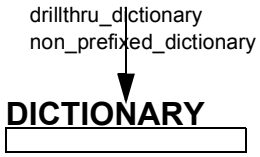
DICTIONARY_SEPARATE_TABLE_INSERT

```
/* Usage */
DICTIONARY_SEPARATE_TABLE_INSERT *
   dictionary_separate_table_insert_new(
      DICTIONARY *post_dictionary->
         original_post_dictionary,
      char *application_name,
      char *login_name,
      LIST *relation_mto1_list,
      LIST *folder_attribute_date_name_list(),
      LIST *folder_attribute_append_isa_list,
      LIST *folder_attribute_name_list );

/* Process */
DICTIONARY_SEPARATE_TABLE_INSERT *
   dictionary_separate_table_insert_calloc(
      void );

if ( !dictionary_length( original_post_dictionary ) )
{
   return this;
}

DICTIONARY_SEPARATE_TRIM_DOUBLE_BRACKET *
   dictionary_separate_trim_double_bracket_new(
      original_post_dictionary );

DICTIONARY_SEPARATE_PARSE *
   dictionary_separate_parse_new(
      DICTIONARY_SEPARATE_SORT_PREFIX,
      DICTIONARY_SEPARATE_QUERY_PREFIX,
      DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
      DICTIONARY_SEPARATE_IGNORE_PREFIX,
      DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
      DICTIONARY_SEPARATE_PAIR_PREFIX,
      ATTRIBUTE_MULTI_KEY_DELIMITER,
      dictionary_separate_trim_double_bracket_new()->
         dictionary );

DICTIONARY_SEPARATE_DATE_CONVERT *
   dictionary_separate_date_convert_prompt_new(
      PROMPT_LOOKUP_FROM_PREFIX,
      PROMPT_LOOKUP_TO_PREFIX,
      application_name,
      login_name,
      folder_attribute_date_name_list,
      dictionary_separate_parse_new() );

DICTIONARY_SEPARATE_SQL_INJECTION_ESCAPE *
   dictionary_separate_sql_injection_escape_new(
      folder_attribute_append_isa_list,
      dictionary_separate_date_convert_prompt_new() );

DICTIONARY *drillthru_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      drillthru_dictionary );

DICTIONARY *ignore_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      ignore_dictionary );

DICTIONARY *non_prefixed_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      non_prefixed_dictionary );

DICTIONARY *prompt_dictionary =
   dictionary_separate_prompt(
      PROMPT_LOOKUP_FROM_PREFIX,
      PROMPT_LOOKUP_TO_PREFIX,
      relation_mto1_list,
      folder_attribute_name_list,
      dictionary_separate_sql_injection_escape_new()->
         non_prefixed_dictionary );

DICTIONARY *
   dictionary_separate_table_insert_query_dictionary(
      prompt_dictionary );

DICTIONARY *pair_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      pair_dictionary );

LIST *ignore_name_list =
   LIST *dictionary_key_list(
      ignore_dictionary );

LIST *prompt_name_list =
   dictionary_key_list(
      prompt_dictionary );

/* Usage */
DICTIONARY *
   dictionary_separate_table_insert_query_dictionary(
      DICTIONARY *prompt_dictionary );

/* Process */
DICTIONARY *query_prompt_query_dictionary(
   QUERY_RELATION_OPERATOR_PREFIX,
   QUERY_EQUAL,
   prompt_dictionary );

/* Attributes */
DICTIONARY *drillthru_dictionary;
DICTIONARY *ignore_dictionary;
DICTIONARY *non_prefixed_dictionary;
DICTIONARY *prompt_dictionary;
DICTIONARY *query_dictionary;
DICTIONARY *pair_dictionary;
LIST *ignore_name_list;
LIST *prompt_name_list;
```



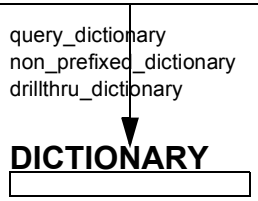
DICTIONARY_SEPARATE_PROMPT_PROCESS

```
/* Usage */
DICTIONARY_SEPARATE_PROMPT_PROCESS *
    dictionary_separate_prompt_process_new(
        DICTIONARY *post_dictionary->
            original_post_dictionary,
        char *application_name,
        char *login_name,
        LIST *prompt_process_date_name_list(),
        LIST *folder_attribute_name_list_attribute_list() );

/* Process */
DICTIONARY_SEPARATE_PROMPT_PROCESS *
    dictionary_separate_prompt_process_calloc(
        void );

if ( !dictionary_length( original_post_dictionary ) )
{
    return this;
}

DICTIONARY_SEPARATE_TRIM_DOUBLE_BRACKET *
    dictionary_separate_trim_double_bracket_new(
        original_post_dictionary );

DICTIONARY_SEPARATE_PARSE *
    dictionary_separate_parse_new(
        DICTIONARY_SEPARATE_SORT_PREFIX,
        DICTIONARY_SEPARATE_QUERY_PREFIX,
        DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
        DICTIONARY_SEPARATE_IGNORE_PREFIX,
        DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
        DICTIONARY_SEPARATE_PAIR_PREFIX,
        ATTRIBUTE_MULTI_KEY_DELIMITER,
        dictionary_separate_trim_double_bracket_new()->
            dictionary );

DICTIONARY_SEPARATE_DATE_CONVERT *
    dictionary_separate_date_convert_prompt_new(
        PROMPT_LOOKUP_FROM_PREFIX,
        PROMPT_LOOKUP_TO_PREFIX,
        application_name,
        login_name,
        prompt_process_date_name_list,
        dictionary_separate_parse_new() );

DICTIONARY_SEPARATE_SQL_INJECTION_ESCAPE *
    dictionary_separate_sql_injection_escape_new(
        folder_attribute_name_list_attribute_list,
        dictionary_separate_date_convert_prompt_new() );

DICTIONARY *non_prefixed_dictionary =
    dictionary_separate_sql_injection_escape_new()->
        non_prefixed_dictionary;

DICTIONARY *drillthru_dictionary =
    dictionary_append_dictionary(
        non_prefixed_dictionary,
        dictionary_separate_sql_injection_escape_new()->
            drillthru_dictionary );

/* Attributes */
DICTIONARY *drillthru_dictionary;
DICTIONARY *non_prefixed_dictionary;
```



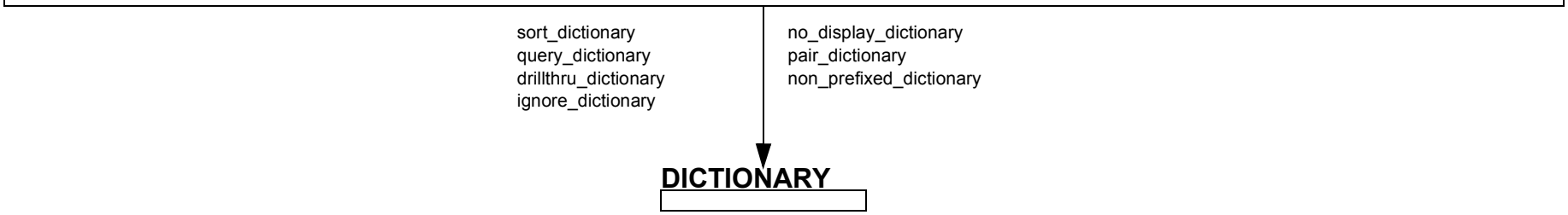
DICTIONARY_SEPARATE_LOOKUP_DELETE

```
/* Usage */
DICTIONARY_SEPARATE_LOOKUP_DELETE *
   dictionary_separate_lookup_delete_new(
      DICTIONARY *post_dictionary->
         original_post_dictionary,
      char *application_name,
      char *login_name,
      LIST *folder_attribute_date_name_list(),
      LIST *folder_attribute_append_isa_list() );

/* Process */
DICTIONARY_SEPARATE_LOOKUP_DELETE *
   dictionary_separate_lookup_delete_calloc(
      void );

if ( !dictionary_length( original_post_dictionary ) )
{
   return this;
}

DICTIONARY_SEPARATE_TRIM_DOUBLE_BRACKET *
   dictionary_separate_trim_double_bracket_new(
      original_post_dictionary );

DICTIONARY_SEPARATE_PARSE *
   dictionary_separate_parse_new(
      DICTIONARY_SEPARATE_SORT_PREFIX,
      DICTIONARY_SEPARATE_QUERY_PREFIX,
      DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
      DICTIONARY_SEPARATE_IGNORE_PREFIX,
      DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
      DICTIONARY_SEPARATE_PAIR_PREFIX,
      ATTRIBUTE_MULTI_KEY_DELIMITER,
      dictionary_separate_trim_double_bracket_new()->
         dictionary );

DICTIONARY_SEPARATE_DATE_CONVERT *
   dictionary_separate_date_convert_prompt_new(
      PROMPT_LOOKUP_FROM_PREFIX,
      PROMPT_LOOKUP_TO_PREFIX,
      application_name,
      login_name,
      folder_attribute_date_name_list,
      dictionary_separate_parse_new() );

DICTIONARY_SEPARATE_SQL_INJECTION_ESCAPE *
   dictionary_separate_sql_injection_escape_new(
      folder_attribute_append_isa_list,
      dictionary_separate_date_convert_prompt_new() );

DICTIONARY *query_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      query_dictionary;

DICTIONARY *non_prefixed_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      non_prefixed_dictionary;

DICTIONARY *drillthru_dictionary =
   dictionary_separate_sql_injection_escape_new()->
      drillthru_dictionary;

/* Attributes */
DICTIONARY *query_dictionary;
DICTIONARY *non_prefixed_dictionary;
DICTIONARY *drillthru_dictionary;
```



DICTIONARY_SEPARATE_DATE_CONVERT (1)

```
/* Usage */
DICTIONARY_SEPARATE_DATE_CONVERT *
    dictionary_separate_date_convert_prompt_new(
        const char *PROMPT_LOOKUP_FROM_PREFIX,
        const char *PROMPT_LOOKUP_TO_PREFIX,
        char *application_name,
        char *login_name,
        LIST *folder_attribute_date_name_list(),
        DICTIONARY_SEPARATE_PARSE *
            dictionary_separate_parse );

/* Process */
DICTIONARY_SEPARATE_DATE_CONVERT *
    dictionary_separate_date_convert_calloc(
        void );

void dictionary_separate_date_convert_prompt_frame(
    PROMPT_LOOKUP_FROM_PREFIX,
    PROMPT_LOOKUP_TO_PREFIX,
    application_name,
    login_name,
    folder_attribute_date_name_list,
    this->non_prefixed_dictionary /* in/out */ );

/* Usage */
DICTIONARY_SEPARATE_DATE_CONVERT *
    dictionary_separate_date_convert_edit_new(
        char *application_name,
        char *login_name,
        LIST *folder_attribute_date_name_list,
        DICTIONARY_SEPARATE_PARSE *
            dictionary_separate_parse );

/* Process */
DICTIONARY_SEPARATE_DATE_CONVERT *
    dictionary_separate_date_convert_calloc();

int list_length( folder_attribute_date_name_list );

if ( list_length() )
{
    int dictionary_highest_index(
        this->non_prefixed_dictionary );

    if ( dictionary_highest_index() )
    {
        dictionary_separate_date_convert_edit_frame(
            application_name,
            login_name,
            folder_attribute_date_name_list,
            this->non_prefixed_dictionary /* in/out */,
            dictionary_highest_index() );
    }
}

/* Attribute */
DICTIONARY *sort_dictionary;
DICTIONARY *query_dictionary;
DICTIONARY *drillthru_dictionary;
DICTIONARY *ignore_dictionary;
DICTIONARY *no_display_dictionary;
DICTIONARY *pair_dictionary;
DICTIONARY *non_prefixed_dictionary;
```



DICTIONARY_SEPARATE_DATE_CONVERT (2)

```
/* Usage */
void dictionary_separate_date_convert_prompt_frame(
    const char *PROMPT_LOOKUP_FROM_PREFIX,
    const char *PROMPT_LOOKUP_TO_PREFIX,
    char *application_name,
    char *login_name,
    LIST *folder_attribute_date_name_list,
    DICTIONARY *dictionary /* in/out */ );

/* Process */
for char *attribute_name in
    folder_attribute_date_name_list
{
    char *dictionary_get( attribute_name, dictionary );

    if ( dictionary_get() )
    {
        (void)dictionary_separate_date_convert(
            dictionary /* in/out */,
            application_name,
            login_name,
            dictionary_get() /* date_string */,
            attribute_name /* key */ );

        continue;
    }

    snprintf(
        key[],
        sizeof ( key ),
        "%s%s",
        PROMPT_LOOKUP_FROM_PREFIX,
        attribute_name );

    char *dictionary_get( key, dictionary );

    if ( dictionary_get() )
    {
        char *strdup( key );

        boolean dictionary_separate_date_convert(
            dictionary /* in/out */,
            application_name,
            login_name,
            dictionary_get() /* date_string */,
            strdup() /* key */ );

        if ( !dictionary_separate_date_convert() )
        {
            free( strdup() );
        }
    }

    snprintf(
        key[],
        sizeof ( key ),
        "%s%s",
        PROMPT_LOOKUP_TO_PREFIX,
        attribute_name );

    char *dictionary_get( key, dictionary );

    if ( dictionary_get() )
    {
        char *strdup( key );

        boolean dictionary_separate_date_convert(
            dictionary /* in/out */,
            application_name,
            login_name,
            dictionary_get() /* date_string */,
            strdup() /* key */ );

        if ( !dictionary_separate_date_convert() )
        {
            free( strdup() );
        }
    }
} /* for each attribute_name */

/* Usage */
void dictionary_separate_date_convert_edit_frame(
    char *application_name,
    char *login_name,
    LIST *folder_attribute_date_name_list,
    DICTIONARY *dictionary /* in/out */,
    int dictionary_highest_index );

/* Process */
for ( int index = 1;
      index <= dictionary_highest_index;
      index++ )
{
    for char *attribute_name in
        folder_attribute_date_name_list
    {
        snprintf(
            key[],
            sizeof ( key ),
            "%s_%d",
            attribute_name,
            index );

        char *dictionary_get( key, dictionary );

        if ( dictionary_get() )
        {
            char *strdup( key );

            boolean dictionary_separate_date_convert(
                dictionary /* in/out */,
                application_name,
                login_name,
                dictionary_get() /* date_string */,
                strdup() /* key */ );

            if ( !dictionary_separate_date_convert() )
            {
                free( strdup() );
            }
        } /* if dictionary_get() */
    } /* for each attribute_name */
} /* for each index */
```

DICTIONARY_SEPARATE_DATE_CONVERT (3)

```
/* Usage */
boolean dictionary_separate_date_convert(
    DICTIONARY *dictionary /* in/out */,
    char *application_name,
    char *login_name,
    char *date_string,
    char *key );

/* Process */
char *dictionary_separate_date_convert_string(
    application_name,
    login_name,
    date_string );

if ( !dictionary_separate_date_convert_string() )
    return 0;

if ( strcmp(
        date_string,
        dictionary_separate_date_convert_string() )
           != 0 )
{
    dictionary_set(
        dictionary,
        key,
        dictionary_separate_date_convert_string() );

    return 1;
}
else
{
    free( dictionary_separate_date_convert_string() );
    return 0;
}

/* Usage */
char *dictionary_separate_date_convert_string(
    char *application_name,
    char *login_name,
    char *date_string );

/* Process */
char string[];
char *ptr = string;

/* Loop if Or , */
for (     int i = 0;
          piece(
             date_piece_string[],
             ',',
             date_string,
             i );
          i++ )
{
    int strlen( date_piece_string );
    if ( strlen() > 20 ) return NULL;

    int string_character_count(
            ' ',
            date_piece_string );

    if ( string_character_count() > 1 ) return NULL;

    DATE_CONVERT *date_convert =
       date_convert_destination_international(
          application_name,
          login_name,
          date_piece_string /* date_string */ );

    if (    date_convert->source_enum ==
            date_convert_unknown )
    {
       date_convert_free( date_convert );
       return NULL;
    }

    if ( i ) ptr += sprintf( ptr, "," );

    ptr += sprintf( ptr,
       "%s",
       date_convert->return_date_string );
}
```

DICTIONARY_SEPARATE_TRIM_DOUBLE_BRACKET

```
/* Usage */
DICTIONARY_SEPARATE_TRIM_DOUBLE_BRACKET *
    dictionary_separate_trim_double_bracket_new(
        DICTIONARY *original_post_dictionary /* in/out */ );

/* Process */
DICTIONARY_SEPARATE_TRIM_DOUBLE_BRACKET *
    dictionary_separate_trim_double_bracket_calloc(
        void );

DICTIONARY *dictionary = original_post_dictionary;

void dictionary_separate_trim_double_bracket(
    dictionary /* in/out */ );

/* Usage */
void dictionary_separate_trim_double_bracket(
    DICTIONARY *dictionary /* in/out */ );

/* Process */
DICTIONARY *dictionary_trim_double_bracked_data(
    dictionary );

/* Attribute */
DICTIONARY *dictionary;
```


DICTIONARY

DICTIONARY_SEPARATE_SQL_INJECTION_ESCAPE

```
/* Usage */
DICTIONARY_SEPARATE_SQL_INJECTION_ESCAPE *
   dictionary_separate_sql_injection_escape_new(
      LIST *folder_attribute_list,
      DICTIONARY_SEPARATE_DATE_CONVERT *
         dictionary_separate_date_convert );

/* Process */
DICTIONARY_SEPARATE_SQL_INJECTION_ESCAPE *
   dictionary_separate_sql_injection_escape_calloc(
      void );

LIST *dictionary_key_list(
   this->non_prefixed_dictionary );

for key in dictionary_key_list()
{
   char *datum =
      dictionary_get(
         key,
         this->non_prefixed_dictionary );

   if ( !datum ) exit( 1 );
```

```
string_trim_index(
   key_trimmed[],
   key );

FOLDER_ATTRIBUTE *folder_attribute =
   folder_attribute_seek(
      (char *)0 /* folder_name */,
      key_trimmed,
      folder_attribute_list );

if ( folder_attribute )
{
   boolean attribute_is_number(
      folder_attribute->
         attribute->
         datatype_name );

   if ( attribute_is_number() )
   {
      /* Trims , and $ */
      (void)attribute_trim_number_characters(
         datum );
   }
}
```

```
   char *security_sql_injection_escape(
      SECURITY_ESCAPE_CHARACTER_STRING,
      datum );

   /* Might be a constant */
   /* free( datum ); */

   dictionary_set(
      this->non_prefixed_dictionary,
      key,
      security_sql_injection_escape() );
}

/* Attributes */
DICTIONARY *sort_dictionary;
DICTIONARY *query_dictionary;
DICTIONARY *drillthru_dictionary;
DICTIONARY *ignore_dictionary;
DICTIONARY *no_display_dictionary;
DICTIONARY *pair_dictionary;
DICTIONARY *non_prefixed_dictionary;
```

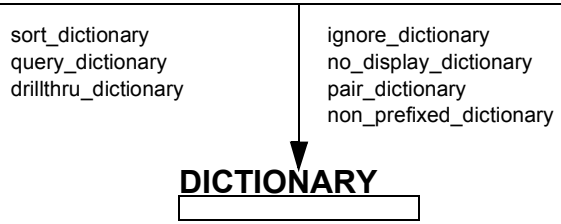
sort_dictionary
query_dictionary
drillthru_dictionary
ignore_dictionary

no_display_dictionary
pair_dictionary
non_prefixed_dictionary

DICTIONARY

DICTIONARY_SEPARATE_PARSE (1)

```
/* Usage */
DICTIONARY_SEPARATE_PARSE *
   dictionary_separate_parse_new(
      const char *
         DICTIONARY_SEPARATE_SORT_PREFIX,
      const char *
         DICTIONARY_SEPARATE_QUERY_PREFIX,
      const char *
         DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
      const char *
         DICTIONARY_SEPARATE_IGNORE_PREFIX,
      const char *
         DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
      const char *
         DICTIONARY_SEPARATE_PAIR_PREFIX,
      const char ATTRIBUTE_MULTI_KEY_DELIMITER,
      DICTIONARY *dictionary );

/* Process */
DICTIONARY_SEPARATE_PARSE *
   dictionary_separate_parse_calloc(
      void );

LIST *dictionary_key_list( dictionary );

if ( !list_length( dictionary_key_list() ) return this;

DICTIONARY *sort_dictionary =
   dictionary_separate_parse_prefix(
      DICTIONARY_SEPARATE_SORT_PREFIX,
      dictionary,
      dictionary_key_list() );

void dictionary_separate_parse_multi(
   ATTRIBUTE_MULTI_KEY_DELIMITER,
   (char *)0 /* QUERY_RELATION_OPERATOR_PREFIX */,
   (char *)0 /* WIDGET_MULTI_ORIGINAL_PREFIX */,
   sort_dictionary /* in/out */ );

DICTIONARY *query_dictionary =
   dictionary_separate_parse_prefix(
      DICTIONARY_SEPARATE_QUERY_PREFIX,
      dictionary,
      dictionary_key_list() );

void dictionary_separate_parse_multi(
   ATTRIBUTE_MULTI_KEY_DELIMITER,
   QUERY_RELATION_OPERATOR_PREFIX,
   WIDGET_MULTI_ORIGINAL_PREFIX,
   query_dictionary /* in/out */ );

DICTIONARY *drillthru_dictionary =
   dictionary_separate_parse_prefix(
      DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
      dictionary,
      dictionary_key_list() );

void dictionary_separate_parse_multi(
   ATTRIBUTE_MULTI_KEY_DELIMITER,
   QUERY_RELATION_OPERATOR_PREFIX,
   drillthru_dictionary /* in/out */ );

DICTIONARY *ignore_dictionary =
   dictionary_separate_parse_prefix(
      DICTIONARY_SEPARATE_IGNORE_PREFIX,
      dictionary,
      dictionary_key_list() );

void dictionary_separate_parse_multi(
   ATTRIBUTE_MULTI_KEY_DELIMITER,
   (char *)0 /* QUERY_RELATION_OPERATOR_PREFIX */,
   ignore_dictionary /* in/out */ );

DICTIONARY *no_display_dictionary =
   dictionary_separate_parse_prefix(
      DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
      dictionary,
      dictionary_key_list() );

void dictionary_separate_parse_multi(
   ATTRIBUTE_MULTI_KEY_DELIMITER,
   (char *)0 /* QUERY_RELATION_OPERATOR_PREFIX */,
   no_display_dictionary /* in/out */ );

DICTIONARY *pair_dictionary =
   dictionary_separate_parse_prefix(
      DICTIONARY_SEPARATE_PAIR_PREFIX,
      dictionary,
      dictionary_key_list() );

void dictionary_separate_parse_multi(
   ATTRIBUTE_MULTI_KEY_DELIMITER,
   (char *)0 /* QUERY_RELATION_OPERATOR_PREFIX */,
   pair_dictionary /* in/out */ );

DICTIONARY *non_prefixed_dictionary =
   dictionary_separate_parse_non_prefixed(
      DICTIONARY_SEPARATE_SORT_PREFIX,
      DICTIONARY_SEPARATE_QUERY_PREFIX,
      DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
      DICTIONARY_SEPARATE_IGNORE_PREFIX,
      DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
      DICTIONARY_SEPARATE_PAIR_PREFIX,
      dictionary,
      dictionary_key_list() );

void dictionary_separate_parse_multi(
   ATTRIBUTE_MULTI_KEY_DELIMITER,
   QUERY_RELATION_OPERATOR_PREFIX,
   non_prefixed_dictionary /* in/out */ );

list_free_container( dictionary_key_list() );

/* Attributes */
DICTIONARY *sort_dictionary;
DICTIONARY *query_dictionary;
DICTIONARY *drillthru_dictionary;
DICTIONARY *ignore_dictionary;
DICTIONARY *no_display_dictionary;
DICTIONARY *pair_dictionary;
DICTIONARY *non_prefixed_dictionary;
```



DICTIONARY_SEPARATE_PARSE (2)

```
/* Usage */
DICTIONARY *dictionary_separate_parse_prefix(
    const char *prefix,
    DICTIONARY *dictionary,
    LIST *dictionary_key_list );

/* Process */
DICTIONARY *destination_dictionary =
    dictionary_small();

for char *key in dictionary_key_list
{
    int string_strncmp( key, prefix );
    if ( string_strncmp() != 0 ) continue;
    char *dictionary_get( key, dictionary );

    snprintf(
        new_key[],
        sizeof ( new_key ),
        "%s",
        key + strlen( prefix ) );

    dictionary_set(
        destination_dictionary,
        strdup( new_key ),
        dictionary_get() );
}
```

```
/* Usage */
DICTIONARY *dictionary_separate_parse_non_prefixed(
    const char *DICTIONARY_SEPARATE_SORT_PREFIX,
    const char *DICTIONARY_SEPARATE_QUERY_PREFIX,
    const char *
        DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
    const char *DICTIONARY_SEPARATE_IGNORE_PREFIX,
    const char *
        DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
    const char *DICTIONARY_SEPARATE_PAIR_PREFIX,
    DICTIONARY *dictionary,
    LIST *dictionary_key_list );

/* Process */
DICTIONARY *non_prefixed_dictionary =
    dictionary_large();

for char *key in dictionary_key_list
{
    int string_strncmp(
        key,
        DICTIONARY_SEPARATE_SORT_PREFIX );
    if ( string_strncmp() == 0 ) continue;

    int string_strncmp(
        key,
        DICTIONARY_SEPARATE_QUERY_PREFIX );
    if ( string_strncmp() == 0 ) continue;

    int string_strncmp(
        key,
        DICTIONARY_SEPARATE_DRILLTHRU_PREFIX );
    if ( string_strncmp() == 0 ) continue;

    int string_strncmp(
        key,
        DICTIONARY_SEPARATE_IGNORE_PREFIX );
    if ( string_strncmp() == 0 ) continue;

    int string_strncmp(
        key,
        DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX );
    if ( string_strncmp() == 0 ) continue;

    int string_strncmp(
        key,
        DICTIONARY_SEPARATE_PAIR_PREFIX );
    if ( string_strncmp() == 0 ) continue;

    char *dictionary_get( key, dictionary );

    dictionary_set(
        non_prefixed_dictionary,
        key,
        dictionary_get() );
}
```

DICTIONARY_SEPARATE_PARSE (3)

```
/* Usage */
void dictionary_separate_parse_multi(
    const char ATTRIBUTE_MULTI_KEY_DELIMITER,
    const char *QUERY_RELATION_OPERATOR_PREFIX,
    DICTIONARY *dictionary /* in/out */ );

/* Process */
LIST *dictionary_key_list( dictionary );

for char *multi_key in dictionary_key_list()
{
    boolean string_character_boolean(
        multi_key,
        ATTRIBUTE_MULTI_KEY_DELIMITER );

    if ( !string_character_boolean() ) continue;

    boolean dictionary_separate_parse_relation_boolean(
        QUERY_RELATION_OPERATOR_PREFIX,
        multi_key );

    if ( dictionary_separate_parse_relation_boolean() )
    {
        dictionary_separate_parse_multi_relation(
            ATTRIBUTE_MULTI_KEY_DELIMITER,
            QUERY_RELATION_OPERATOR_PREFIX,
            dictionary /* in/out */,
            multi_key );
    }
    else
    {
        dictionary_separate_parse_multi_non_relation(
            ATTRIBUTE_MULTI_KEY_DELIMITER,
            dictionary /* in/out */,
            multi_key );
    }
}
```

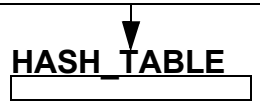
```
/* Usage */
void dictionary_separate_parse_multi_non_relation(
    const char ATTRIBUTE_MULTI_KEY_DELIMITER,
    DICTIONARY *dictionary /* in/out */,
    char *multi_key );

/* Process */
int string_index( multi_key );

(void)string_trim_index(
    multi_key_without_index[]
        /* destination */,
    multi_prefix );

char *dictionary_get(
        multi_key,
        dictionary );

if ( !dictionary_get() || !*dictionary_get() ) return;

void dictionary_separate_parse_multi_key(
    ATTRIBUTE_MULTI_KEY_DELIMITER,
    dictionary /* in/out */,
    string_index(),
    multi_key_without_index /* stack memory */,
    dictionary_get() );

/* Usage */
/*
multi_key=single_datum looks like:
relation_operator_full_name^street_address=equals
*/
void dictionary_separate_parse_multi_relation(
    const char ATTRIBUTE_MULTI_KEY_DELIMITER,
    const char *QUERY_RELATION_OPERATOR_PREFIX,
    DICTIONARY *dictionary /* in/out */,
    char *multi_key );

/* Process */
/* Should be “equals” */
char *single_datum =
    dictionary_get(
        multi_key,
        dictionary );

strcpy(
    key_without_prefix[],
    multi_key +
    strlen(
        QUERY_RELATION_OPERATOR_PREFIX ) );

LIST *attribute_key_list =
    list_string_to_list(
        key_without_prefix,
        ATTRIBUTE_MULTI_KEY_DELIMITER );

for char *single_key in attribute_key_list
{
    snprintf(
        relation_key[],
        sizeof ( relation_key ),
        “%s%s”,
        QUERY_RELATION_OPERATOR_PREFIX,
        single_key );

    dictionary_set(
        dictionary,
        strdup( relation_key ),
        single_datum );
}

list_free( attribute_key_list );
```

DICTIONARY_SEPARATE_PARSE (4)

```
/* Usage */
void dictionary_separate_parse_multi_key(
    const char ATTRIBUTE_MULTI_KEY_DELIMITER,
    DICTIONARY *dictionary /* in/out */,
    int string_index,
    char *multi_key_without_index,
    char *multi_data );

/* Process */
LIST *attribute_key_list =
    list_string_to_list(
        multi_key_without_index,
        ATTRIBUTE_MULTI_KEY_DELIMITER );

LIST *attribute_data_list =
    list_string_to_list(
        multi_data,
        ATTRIBUTE_MULTI_KEY_DELIMITER );

list_rewind( attribute_data_list );

for char *single_key in attribute_key_list
{
    char *single_datum =
        list_get(
            attribute_data_list );

    char *dictionary_separate_parse_multi_index_key(
        string_index,
        single_key );

    dictionary_set(
        dictionary,
        dictionary_separate_parse_multi_index_key(),
        single_datum );

    /* May be QUERY_IS_NULL or QUERY_NOT_NULL */
    if ( !list_last_boolean( attribute_data_list ) )
        list_next( attribute_data_list );
}

list_free_string_list( attribute_key_list );
list_free_container( attribute_data_list );

/* Usage */
char *dictionary_separate_parse_multi_index_key(
    int string_index,
    char *single_key );

/* Process */
if ( string_index > 0 )
{
    snprintf(
        index_key[],
        sizeof ( index_key ),
        "%s_%d",
        single_key,
        string_index );
}
else
{
    string_strcpy(
        index_key[],
        single_key,
        sizeof ( index_key ) );
}

/* Usage */
boolean dictionary_separate_parse_relation_boolean(
    char *QUERY_RELATION_OPERATOR_PREFIX,
    char *multi_key );

/* Process */
if ( !QUERY_RELATION_OPERATOR_PREFIX ) return 0;

int string_strncmp(
    multi_key,
    QUERY_RELATION_OPERATOR_PREFIX );

( string_strncmp() == 0 );
```

DICTIONARY (1)

```
/* Usage */
DICTIONARY *dictionary_new(
    void  );

DICTIONARY *dictionary_small(
    void );

DICTIONARY *dictionary_medium(
    void );

DICTIONARY *dictionary_large(
    void );

DICTIONARY *dictionary_huge(
    void );

DICTIONARY *dictionary_super(
    void );

DICTIONARY *dictionary_duper(
    void );

DICTIONARY *dictionary_string_new(
    char *dictionary_string );

DICTIONARY *dictionary_string_resolve(
    char *dictionary_string );

DICTIONARY *dictionary_index_string2dictionary(
    char *dictionary_string );

DICTIONARY *dictionary_string_dictionary(
    char *dictionary_string );

DICTIONARY *dictionary_string2dictionary(
    char *dictionary_string );

/* Process */
DICTIONARY *dictionary_calloc( void );

/* Usage */
DICTIONARY *dictionary_string_new(
    const char *
        DICTIONARY_ATTRIBUTE_DATUM_DELIMITER
            /* probably "@=" */,
    const char *
        DICTIONARY_ELEMENT_DELIMITER
            /* probably "@&" */,
    char *dictionary_string );

/* Process */
DICTIONARY *dictionary_large();

for (  int i = 0;
        piece_string(
            element[],
            DICTIONARY_ELEMENT_DELIMITER,
            dictionary_string,
            i );
        i++ )
{
    piece_string(
        attribute[],
        DICTIONARY_ATTRIBUTE_DATUM_DELIMITER,
        element,
        0 );

    piece_string(
        datum[],
        DICTIONARY_ATTRIBUTE_DATUM_DELIMITER,
        element,
        1 );

    if ( *datum )
    {
        dictionary_set(
            dictionary_large(),
            strdup( attribute ),
            strdup( datum ) );
}

/* Attribute */
HASH_TABLE *hash_table;
```



DICTIONARY (2)

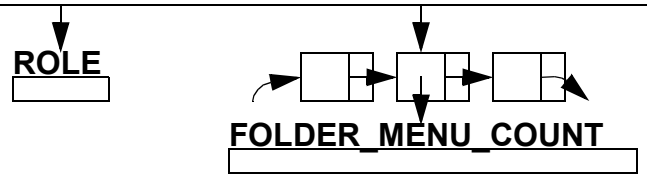
```
/* Usage */
void dictionary_replace_command_line(
    char *command_line,
    DICTIONARY *dictionary );

/* Process */
int dictionary_highest_index(
    DICTIONARY *dictionary );

LIST *column_list( command_line );

if ( list_length( column_list() ) >= 1 ) return;

/* Skip the executable */
list_next( column_list() );

for char *column in column_list()
{
    char *dictionary_get( column, dictionary );

    if ( dictionary_get() )
    {
        string_replace_command_line(
            command_line,
            dictionary_get() /* string */,
            column /* placeholder */ );
    }
    else
    if ( dictionary_highest_index() )
    {
        LIST *dictionary_indexed_data_list(
            DICTIONARY *dictionary,
            int dictionary_highest_index(),
            char *column /* key_prefix */ );

        if ( list_length( dictionary_indexed_data_list() ) )
        {
            char *list_delimited_string(
                dictionary_indexed_data_list(),
                SQL_DELIMITER );

            string_replace_command_line(
                command_line,
                list_delimited_string(),
                column /* placeholder */ );
        }
    }
}

/* Usage */
void dictionary_index_set(
    DICTIONARY *dictionary /* out */,
    DICTIONARY *multi_row_dictionary,
    LIST *dictionary_key_list(),
    int index );

/* Process */
for char *key in dictionary_key_list
{
    int string_index( key );

    if ( index == string_index() )
    {
        char *dictionary_get(
            key,
            multi_row_dictionary );

        dictionary_set(
            dictionary,
            key,
            dictionary_get() /* data */ );
    }
}

/* Usage */
LIST *dictionary_index_populated_name_list(
    LIST *folder_attribute_non_primary_name_list,
    DICTIONARY *multi_row_dictionary,
    int index );

/* Process */
LIST *name_list = list_new();

for char *non_primary_name in
    folder_attribute_non_primary_name_list
{
    snprintf(
        key_index[],
        sizeof ( key_index ),
        "%s_%d",
        non_primary_name,
        index );

    char *dictionary_get(
        key_index,
        multi_row_dictionary )

    if ( dictionary_get() )
    {
        list_set(
            name_list,
            non_primary_name );
    }
}

/* Usage */
void dictionary_prefixed_dictionary_set(
    DICTIONARY *dictionary /* out */,
    char *prefix,
    DICTIONARY *source_dictionary );

/* Process */
LIST *dictionary_key_list( source_dictionary );

for char *key in dictionary_key_list()
{
    char *dictionary_get(
        key,
        source_dictionary );

    char *dictionary_prefixed_key(
        char *prefix,
        char *key );

    dictionary_set(
        dictionary,
        dictionary_prefixed_key(),
        dictionary_get() );
}

list_free_container( dictionary_key_list() );
```

```
/* Usage */
void dictionary_set(
    DICTIONARY *dictionary,
    char *key,
    char *data );

/* Process */

/* Usage */
char *dictionary_get(
    char *key,
    DICTIONARY *dictionary );

/* Process */

/* Usage */
char *dictionary_list_get(
    DICTIONARY *dictionary,
    LIST *key_list );

/* Process */
for char *key in key_list
{
    DICTIONARY *dictionary_get(
        key,
        dictionary );

    if ( dictionary_get() ) return dictionary_get();
}

/* Usage */
char *dictionary_index_get(
    char *key,
    DICTIONARY *dictionary,
    int index );

/* Process */
char *dictionary_attribute_name_append_row_number(
    key /* attribute_name */,
    index /* row_number */ );

char *dictionary_get(
    dictionary_attribute_name_append_row_number(),
    dictionary );

/* Usage */
int dictionary_length(
    DICTIONARY *dictionary );

/* Process */

/* Usage */
LIST *dictionary_key_list(
    DICTIONARY *dictionary );

/* Process */

/* Usage */
LIST *dictionary_ordered_key_list(
    DICTIONARY *dictionary );

/* Usage */
LIST *dictionary_data_list(
    LIST *key_list,
    DICTIONARY *dictionary );

/* Process */

/* Usage */
DICTIONARY *dictionary_row_fetch(
    LIST *attribute_name_list,
    char *appaserver_table_name,
    char *where );

/* Process */
char *dictionary_select_attribute_string(
    LIST *attribute_name_list );

char *dictionary_system_string(
    char *dictionary_select_attribute_string(),
    char *appaserver_table_name,
    char *where );

char *string_pipe_input(
    dictionary_system_string() );

DICTIONARY *dictionary_parse(
    char SQL_DELIMITER,
    LIST *attribute_name_list,
    char *string_pipe_input() );

/* Usage */
DICTIONARY *dictionary_name_list_fetch(
    char *system_string,
    LIST *name_list,
    char delimiter );

/* Process */
char *string_pipe( system_string );

if ( string_pipe() )
{
    DICTIONARY *dictionary_small();
    list_rewind( name_list );

    for (int p = 0;
         piece( datum[], delimiter, string_pipe(), p );
         p++ )
    {
        if ( *datum )
        {
            dictionary_set(
                dictionary_small(),
                list_get( name_list ),
                strdup( datum ) );
        }
        list_next( name_list );
    }
}
```

```
/* Usage */
/* Returns >= 0 */
int dictionary_highest_index(
    DICTIONARY *dictionary );

/* Process */

/* Usage */
int dictionary_key_list_highest_index(
    const char ATTRIBUTE_MULTI_KEY_DELIMITER,
    LIST *key_list,
    DICTIONARY *dictionary,
    int dictionary_highest_index() );

/* Process */

/* Usage */
/* Two notes:
1) No memory gets allocated. Both the key and data
still exist in the append_dictionary dictionary.
2) If a they share a key, then it doesn't clobber the
source_destinatation_dictionary. */

void dictionary_append_dictionary(
    DICTIONARY *source_destination_dictionary,
    DICTIONARY *append_dictionary );

/* Process */

/* Usage */
char *dictionary_display_delimited(
    DICTIONARY *dictionary,
    const char *
        DICTIONARY_ATTRIBUTE_DATUM_DELIMITER,
    const char *
        DICTIONARY_ELEMENT_DELIMITER );

/* Process */

/* Public */
DICTIONARY *dictionary_prefix_extract(
    const char *prefix,
    DICTIONARY *dictionary );

DICTIONARY *dictionary_remove_prefix(
    const char *prefix,
    DICTIONARY *dictionary );

LIST *dictionary_prefix_key_list(
    const char *prefix,
    DICTIONARY *dictionary );

DICTIONARY *dictionary_copy(
    DICTIONARY *dictionary );

DICTIONARY *dictionary_large_copy(
    DICTIONARY *dictionary );

DICTIONARY *dictionary_file_fetch(
    char *filename,
    char delimiter );

void dictionary_free(
    DICTIONARY *dictionary );

void dictionary_free_container(
    DICTIONARY *dictionary );

void dictionary_free_data(
    DICTIONARY *dictionary,
    LIST *dictionary_key_list() );
```

FOLDER_MENU

```
/* Usage */
FOLDER_MENU *folder_menu_fetch(
    char *application_name,
    char *session_key,
    char *role_name,
    char *appaserver_parameter->data_directory
        /* appaserver_data_directory */,
    boolean folder_menu_remove_boolean );

/* Process */
FOLDER_MENU *folder_menu_calloc(
    void );

LIST *folder_menu_lookup_folder_name_list(
    role_name );

LIST *folder_menu_insert_folder_name_list(
    role_name );

ROLE *role =
    role_fetch(
        role_name,
        0 /* not fetch_role_attribute_exclude_list */ );

if ( role->folder_count )
{
    char *folder_menu_filename(
        application_name,
        session_key,
        role_name,
        appaserver_data_directory );

    if ( folder_menu_remove_boolean )
    {
        folder_menu_remove_file(
            folder_menu_filename() );

    LIST *folder_menu_count_list(
        application_name,
        folder_menu_lookup_folder_name_list(),
        folder_menu_filename() );
}

/* Usage */
LIST *folder_menu_lookup_folder_name_list(
    char *role_name );

/* Process */
LIST *role_folder_lookup_list(
    role_name );

LIST *role_folder_name_list(
    role_folder_lookup_list() );

/* Usage */
LIST *folder_menu_insert_folder_name_list(
    char *role_name );

/* Process */
LIST *role_folder_insert_list(
    role_name );

LIST *role_folder_name_list(
    role_folder_lookup_list() );

/* Usage */
char *folder_menu_filename(
    const char *FOLDER_MENU_EXTENSION
        /* probably "tab" */,
    char *application_name,
    char *session_key,
    char *role_name,
    char *appaserver_data_directory );

/* Process */

/* Usage */
void folder_menu_remove_file(
    char *folder_menu_filename() );

/* Process
void timlib_remove_file(
    folder_menu_filename );

/* Attributes */
ROLE *role;
LIST *lookup_folder_name_list;
LIST *insert_folder_name_list;
char *filename;
LIST *count_list;
```



FOLDER_MENU_COUNT

```
/* Usage */
LIST *folder_menu_count_list(
    char *application_name,
    LIST *folder_menu_lookup_folder_name_list(),
    char *folder_menu_filename() );

/* Process */
if ( !timlib_file_exists( folder_menu_filename ) )
{
    void list_write(
        folder_menu_lookup_folder_name_list,
        folder_menu_filename );
}

char *folder_menu_count_system_string(
    application_name,
    folder_menu_filename );

FILE *appaserver_input_pipe(
    folder_menu_count_system_string() );

LIST *list = list_new();

while ( string_input( input, appaserver_input_pipe() )
{
    list_set(
        list,
        folder_menu_count_parse(
            input ) );
}
pclose( appaserver_input_pipe() );

/* Usage */
char *folder_menu_count_system_string(
    char *application_name,
    char *folder_menu_filename() );

/* Process */
sprintf(
    static system_string[],
    "cat %s | table_count.e %s",
    folder_menu_filename,
    application_name );

/* Usage */
FOLDER_MENU_COUNT *folder_menu_count_parse(
    char *input );

/* Process */
piece( char folder_name[], '=', input, 0 );
piece( char count_string[], '=', input, 1 );

FOLDER_MENU_COUNT *folder_menu_count_new(
    strdup( folder_name ),
    count_string );

/* Usage */
FOLDER_MENU_COUNT *folder_menu_count_new(
    char *folder_name,
    char *count_string );

/* Process */
FOLDER_MENU_COUNT *folder_menu_count_calloc(
    void );

unsigned long count =
    strtoul( count_string, (char **)0, 0 );

char *folder_menu_count_display(
    unsigned long count );

/* Usage */
FOLDER_MENU_COUNT *folder_menu_count_seek(
    char *folder_name,
    LIST *folder_menu_count_list() );

/* Process */

/* Attributes */
char *folder_name;
unsigned long count;
char *display;
```

APPLICATION (1)

```
/* Usage */
APPLICATION *application_fetch(
    char *application_name );

/* Process */
char *application_table_name(
    “application” /* folder_name */ );

char *application_primary_where(
    char *application_name );

char *application_system_string(
    char *APPLICATION_SELECT,
    char *application_table_name(),
    char *application_primary_where() );

APPLICATION *application_parse(
    input );

/* Usage */
APPLICATION *application_parse(
    char *input );

/* Process */
APPLICATION *application_new(
    strdup( application_name ) );

/* Usage */
APPLICATION *application_new(
    char *application_name );

/* Process */
APPLICATION *application_calloc(
    void );

/* Attributes */
char *application_name;
char *application_title;
char *user_date_format;
char *relative_source_directory;
int next_session_number;
int next_reference_number;
char *background_color;
char *grace_home_directory;
char *grace_execution_directory;
boolean menu_horizontal_boolean;
boolean ssl_support_boolean;
int max_query_rows_for_drop_downs;
int max_drop_down_size;
```

APPLICATION (2)

```
/* Usage */
char *application_document_root(
    char *application_name,
    char *appaserver_parameter->document_root );

/* Process */
char *application_appaserver_document_root(
    char *document_root );

sprintf( static char directory[],
         "%s/%s",
         application_appaserver_document_root(),
         application_name );

/* Usage */
char *application_document_root_src(
    char *application_name,
    char *appaserver_parameter->document_root );

/* Process */
char *application_appaserver_document_root(
    document_root );

sprintf( static char directory[],
         "%s/src_%s",
         application_appaserver_document_root(),
         application_name );

/* Usage */
char *application_data_directory(
    char *application_name,
    char *appaserver_parameter->data_directory );

/* Process */
sprintf( static char directory[],
         "%s/%s",
         data_directory,
         application_name );

/* Usage */
char *application_appaserver_home_src(
    char *application_name,
    char *mount_point );

/* Process */

/* Usage */
boolean application_name_invalid(
    char *SECURITY_ESCAPE_CHARACTER_STRING,
    char *application_name );

/* Process */
if (!application_name
||  !*application_name
||  string_character_exists( application_name, ' ' )
||  isdigit( *application_name ) )
{
    return 1;
}

for char ch in SECURITY_ESCAPE_CHARACTER_STRING
{
    if ( string_character_exists(
        application_name,
        ch ) )
    {
        return 1;
    }
}

return 0;

/* Usage */
boolean application_exists(
    char *application_log_filename() );

/* Process */
boolean application_file_exist(
    application_log_filename );

/* Usage */
boolean application_menu_horizontal_boolean(
    char *application_name );

/* Process */

/* Usage */
char *application_table_name(
    char *folder_name );

/* Process */
if ( strcmp( folder_name, "application" ) == 0 )
{
    strcpy(
        static table_name[],
        "appaserver_application" );
}
else
if ( strcmp( folder_name, "table" ) == 0 )
{
    strcpy(
        static table_name[],
        "appaserver_table" );
}
else
if ( strcmp( folder_name, "column" ) == 0 )
{
    strcpy(
        static table_name[],
        "appaserver_column" );
}
else
{
    strcpy(
        static table_name[],
        folder_name );
}
```

```
/* Usage */
LIST *application_name_list(
    char *APPLICATION_LOG_EXTENSION,
    char *appaserver_parameter->log_directory );

/* Process */
sprintf(
    regular_expression[],
    “appaserver_*\\.%s”,
    APPLICATION_LOG_EXTENSION );

sprintf(
    system_string[],
    “cd %s && “
    “ls -1 %s |”
    “sed ‘s/^appaserver_//’ |”
    “piece.e ‘.’ 0”,
    log_directory,
    regular_expression );

LIST *list_pipe_fetch( system_string );

/* Usage */
LIST *application_table_name_list( void );

/* Process */
LIST *list_pipe_fetch( “echo \”show tables;\” | sql.e” );

/* Usage */
LIST *application_non_system_folder_name_list(
    void );

/* Process */
LIST *list_pipe_fetch( “application_folder_list.sh n” );

/* Usage */
char *application_email_file_system_string(
    char *subject,
    char *filename,
    char *email_address );

/* Process */
sprintf( system_string,
    “mutt -s ‘%s’ -a %s %s”,
    subject,
    filename,
    email_address );

/* Usage */
char *application_server_address( void );

/* Process */
char *string_pipe_fetch( “ip_address.sh” );

/* Usage */
char *application_http_prompt(
    char *appaserver_parameter->cgi_directory,
    char *application_server_address(),
    boolean application->ssl_support_boolean );

/* Process */
char *application_http_prefix(
    ssl_support_boolean );

sprintf(static http_prompt[],
        “%s://%s%s”,
        application_http_prefix(),
        application_server_address,
        cgi_directory );

/* Usage */
LIST *application_relative_source_directory_list(
    char *application_name );

/* Process */
LIST *list = list_new();

char *application_relative_source_directory(
    application_name );

for(  int p = 0;
      piece(
          char source_directory[],
          ‘:’,
          application_relative_source_directory(),
          p );
      p++ )
{
   list_set(
      list,
      strdup( source_directory ) );
}

/* Usage */
LIST *application_not_null_data_list(
     char *table_name,
     char *column_name );

/* Process */
char *appliction_not_null_system_string(
     char *table_name,
     char *column_name );

LIST *list_pipe_fetch(
     application_not_null_system_string() );
```

APPLICATION (4)

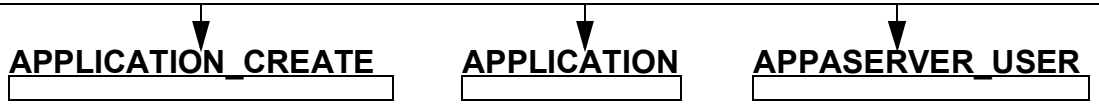
```
/* Public */
char *application_title(
    char *application_name );

char *application_relative_source_directory(
    char *application_name );

char *application_background_color(
    char *application_name );

char *application_grace_home_directory(
    char *application_name );

char *application_grace_execution_directory(
    char *application_name );

boolean application_menu_horizontal_boolean(
    char *application_name );

boolean application_ssl_support_boolean(
    char *application_name );

int application_max_query_rows_for_drop_downs(
    char *application_name );

int application_max_drop_down_size(
    char *application_name );

char *application_first_relative_source_directory(
    char *application_name );

char *application_http_prefix(
    char *application_name );

void application_output_ftp_prompt(
    char *prompt_filename,
    char *prompt,
    char *target,
    char *mime_type );
```

APPLICATION_CLONE (1)

```
/* Usage */
APPLICATION_CLONE *application_clone_new(
    char *current_application_name,
    char *login_name,
    char *sql_injection_escape(
        destination_application_name ),
    char *application_clone_sql_gz_filespecification(),
    char *sql_injection_escape(
        application_title ),
    boolean application_clone_post_signup_boolean(),
    char *appaserver_parameter->document_root,
    char *appaserver_parameter->data_directory,
    char *appaserver_parameter->upload_directory,
    char *appaserver_parameter->log_directory,
    char *appaserver_parameter->mount_point );

/* Process */
APPLICATION_CLONE *application_clone_calloc(
    void );

APPLICATION_CREATE *application_create =
    application_create_new(
        destination_application_name,
        application_title,
        document_root,
        data_directory,
        upload_directory,
        log_directory,
        mount_point );

if (    application_create->application_name_invalid
||      application_create->
            application_log_exists_boolean )
{
    return this;
}

if ( application_clone_sql_gz_filespecification )
{
    char *application_clone_gz_system_string(
        char *destination_application_name,
        char *application_clone_sql_gz_filespecification );
}
else
{
    LIST *application_clone_zaptable_list();

    char *application_clone_database_system_string(
        current_application_name,
        destination_application_name,
        application_clone_zaptable_list() );
}

APPLICATION *application =
    application_fetch(
        current_application_name );

char *application_clone_application_system_string(
    APPLICATION_TABLE,
    destination_application_name,
    application_title,
    application->ssl_support_boolean );

boolean entity_contact_key_boolean(
    ENTITY_TABLE,
    ENTITY_CONTACT_KEY_COLUMN );

APPASERVER_USER *
    application_clone_appaserver_user(
        login_name,
        application_clone_post_signup_boolean,
        entity_contact_key_boolean() );

char *application_clone_insert_user_system_string(
    APPASERVER_USER_TABLE,
    destination_application_name,
    application_clone_appaserver_user()->
        full_name,
    application_clone_appaserver_user()->
        contack_key,
    application_clone_appaserver_user()->
        login_name,
    application_clone_appaserver_user()->
        password,
    application_clone_appaserver_user()->
        user_date_format,
    entity_contact_key_boolean() );

char *application_clone_insert_role_system_string(
    ROLE_SYSTEM,
    ROLE_SUPERVISOR,
    destination_application_name,
    application_clone_appaserver_user()->
        full_name,
    application_clone_appaserver_user()->
        contact_key );
    entity_contact_key_boolean() );

char *application_clone_insert_default_system_string(
    LOGIN_DEFAULT_ROLE_TABLE,
    ROLE_SUPERVISOR,
    destination_application_name,
    application_clone_appaserver_user()->
        full_name,
    application_clone_appaserver_user()->
        contact_key,
    entity_contact_key_boolean() );

char *application_clone_insert_entity_system_string(
    ENTITY_TABLE,
    destination_application_name,
    application_clone_appaserver_user()->
        full_name,
    application_clone_appaserver_user()->
        contact_key,
    entity_contact_key_boolean() );

char *insert_self_system_string =
    application_clone_insert_entity_system_string(
        ENTITY_SELF_TABLE,
        destination_application_name,
        application_clone_appaserver_user()->
            full_name,
        application_clone_appaserver_user()->
            contact_key,
        entity_contact_key_boolean() );
```

APPLICATION_CLONE (2)

```
/* Attributes */
APPLICATION_CREATE *application_create;
char *gz_system_string;
LIST *zaptable_list;
char *database_system_string;
APPLICATION *application;
char *application_system_string;
boolean entity_contact_key_boolean;
APPASERVER_USER *appaserver_user;
char *insert_user_system_string;
char *insert_role_system_string;
char *insert_default_system_string;
char *insert_entity_system_string;
char *insert_self_system_string;
```



APPLICATION_CLONE (3)

```
/* Usage */
char *application_clone_insert_user_system_string(
    const char *APPASERVER_USER_TABLE,
    char *destination_application_name,
    char *full_name,
    char *contact_key,
    char *login_name,
    char *password,
    char *user_date_format,
    boolean entity_contact_key_boolean );

/* Process */
if ( entity_contact_key_boolean )
{
    char *field =
        "appaserver_full_name,"
        "appaserver_contact_key,"
        "login_name,"
        "password,"
        "user_date_format";

    snprintf(
        system_string[],
        sizeof ( system_string ),
        "echo \"%s^%s^%s^%s^%s\" | "
        "insert_statement.e "
        "table=%s "
        "field=%s "
        "delimiter='^' | "
        "sql.e %s",
        full_name,
        contact_key,
        login_name,
        password,
        user_date_format,
        APPASERVER_USER_TABLE,
        field,
        destination_application_name );
}
else
{
    char *field =
        "appaserver_full_name,"
        "login_name,"
        "password,"
        "user_date_format";

    snprintf(
        system_string[],
        sizeof ( system_string ),
        "echo \"%s^%s^%s^%s\" | "
        "insert_statement.e "
        "table=%s "
        "field=%s "
        "delimiter='^' | "
        "sql.e %s",
        full_name,
        login_name,
        password,
        user_date_format,
        APPASERVER_USER_TABLE,
        field,
        destination_application_name );
}
```

APPLICATION_CLONE (4)

```
/* Usage */
char *application_clone_insert_role_system_string(
    const char *ROLE_SYSTEM,
    const char *ROLE_SUPERVISOR,
    char *destination_application_name,
    char *full_name,
    char *contact_key,
    boolean entity_contact_key_boolean );

/* Process */
char system_string[];
char *ptr = system_string;

if ( entity_contact_key_boolean )
{
    char *field =
        "appaserver_full_name,"
        "appaserver_contact_key,"
        "role";

    ptr += sprintf( ptr,
        "echo \"%s^%s^%s\" | "
        "insert_statement.e "
        "table=role_appaserver_user "
        "field=%s "
        "delimiter='^' | "
        "sql.e %s && ",
        full_name,
        contact_key,
        ROLE_SYSTEM,
        field,
        destination_appliction_name );

    ptr += sprintf( ptr,
        "echo \"%s^%s^%s\" | "
        "insert_statement.e "
        "table=role_appaserver_user "
        "field=%s "
        "delimiter='^' | "
        "sql.e %s",
        full_name,
        contact_key,
        ROLE_SUPERVISOR,
        field,
        destination_application_name );
}
else
{
    char *field =
        "appaserver_full_name,"
        "role";

    ptr += sprintf( ptr,
        "echo \"%s^%s\" | "
        "insert_statement.e "
        "table=role_appaserver_user "
        "field=%s "
        "delimiter='^' | "
        "sql.e %s && ",
        full_name,
        ROLE_SYSTEM,
        field,
        destination_appliction_name );

    ptr += sprintf( ptr,
        "echo \"%s^%s\" | "
        "insert_statement.e "
        "table=role_appaserver_user "
        "field=%s "
        "delimiter='^' | "
        "sql.e %s",
        full_name,
        ROLE_SUPERVISOR,
        field,
        destination_application_name );
}
```

APPLICATION_CLONE (5)

```
/* Usage */
char *application_clone_insert_default_system_string(
    const char *LOGIN_DEFAULT_ROLE_TABLE,
    const char *ROLE_SUPERVISOR,
    char *destination_application_name,
    char *full_name,
    char *contact_key,
    boolean entity_contact_key_boolean );

/* Process */
if ( entity_contact_key_boolean )
{
    char *field =
        "appaserver_full_name,"
        "appaserver_contact_key,"
        "role";

    snprintf(
        system_string[],
        sizeof ( system_string ),
        "echo \"%s^%s^%s\" | "
        "insert_statement.e "
        "table=%s "
        "field=%s "
        "delimiter='^' | "
        "sql.e %s",
        full_name,
        contact_key,
        ROLE_SUPERVISOR,
        LOGIN_DEFAULT_ROLE_TABLE,
        field,
        destination_application_name );
}
else
{
    char *field =
        "appaserver_full_name,"
        "appaserver_contact_key,"
        "role";

    snprintf(
        system_string[],
        sizeof ( system_string ),
        "echo \"%s^%s\" | "
        "insert_statement.e "
        "table=%s "
        "field=%s "
        "delimiter='^' | "
        "sql.e %s",
        full_name,
        ROLE_SUPERVISOR,
        LOGIN_DEFAULT_ROLE_TABLE,
        field,
        destination_application_name );
}

/* Usage */
char *application_clone_insert_entity_system_string(
    const char *table_name,
    char *destination_application_name,
    char *full_name,
    char *contact_key,
    boolean entity_contact_key_boolean );

/* Process */
if ( entity_contact_key_boolean )
{
    char *field =
        "full_name,"
        "contact_key";

    snprintf(
        system_string[],
        sizeof ( system_string ),
        "echo \"%s^%s\" | "
        "insert_statement.e "
        "table=%s "
        "field=%s "
        "delimiter='^' | "
        "sql.e %s",
        full_name,
        contact_key,
        table_name,
        field,
        destination_application_name );
}
else
{
    char *field =
        "appaserver_full_name,"
        "appaserver_contact_key";

    snprintf(
        system_string[],
        sizeof ( system_string ),
        "echo \"^%s\" | "
        "insert_statement.e "
        "table=%s "
        "field=%s "
        "delimiter='^' | "
        "sql.e %s",
        full_name,
        table_name,
        field,
        destination_application_name );
}
```

APPLICATION_CLONE (6)

```
/* Usage */
/* Mutually exclusive */
char *application_clone_database_system_string(
    char *current_application_name,
    char *destination_application_name,
    LIST *application_clone_zaptable_list() );

/* Process */
char *list_display_delimited(
    application_clone_zaptable_list,
    ' ' );

sprintf( system_string[],
    "mysqldump_clone.sh %s %s %s",
    current_application_name,
    destination_application_name,
    list_display_delimiter() );

/* Usage */
/* Mutually exclusive */
char *application_clone_sql_gz_filespecification(
    const char *APPLICATION_CLONE_SQL_GZ,
    char *mount_point );

/* Process */
snprintf(
    static filenspecification[],
    sizeof ( filespecification ),
    "%s/%s",
    mount_point,
    APPLICATION_CLONE_SQL_GZ );

/* Usage */
char *application_clone_application_system_string(
    const char *APPLICATION_TABLE,
    char *destination_application_name,
    char *application_title,
    boolean ssl_support_boolean );

/* Process */
snprintf(
    update_statement[],
    sizeof ( update_statement ),
    "update %s set "
    "application='%s',"
    "application_title='%s',"
    "ssl_support_yn='%c';",
    APPLICATION_TABLE,
    destination_appliction_name,
    application_title,
    (ssl_support_boolean)
        ? 'y'
        : 'n' );

snprintf(
    system_string[],
    sizeof ( system_string ),
    "echo \"%s\" | sql.e %s",
    update_statement,
    destination_application_name );

/* Usage */
APPASERVER_USER *
    application_clone_appaserver_user(
        char *login_name,
        boolean application_clone_post_signup_boolean,
        boolean entity_contact_key_boolean );

/* Process */
if ( !application_clone_post_signup_boolean )
{
    APPASERVER_USER *appaserver_user_login_fetch(
        login_name,
        0 /* not fetch_role_name_list */ );

    return this;
}

APPASERVER_USER *appaserver_user_new(
    "Fullname Placeholder" /* full_name */ );

if ( entity_contact_key_boolean )
{
    appaserver_user_new()->contact_key =
        "1234 Set Me" /* street_address */ );
}

appaserver_user_new()->login_name = login_name;

char *security_generate_password();

appaserver_user_new()->password =
    security_generate_password();

appaserver_user_new()->user_date_format =
    "american";

/* Usage */
LIST *application_clone_zaptable_list(
    void );

/* Process */
LIST *list = list_new();

list_set( list, "login_default_role" );
list_set( list, "role_appaserver_user" );
list_set( list, "self" );
list_set( list, "entity" );
list_set( list, "appaserver_user" );
list_set( list, "session );

/* Usage */
void application_clone_system(
    boolean execute_boolean,
    char *system_string );

/* Process */
if ( execute_boolean )
    system( system_string );
else
    printf( "<p>%s\n", system_string );

/* Usage */
boolean application_clone_post_signup_boolean(
    char *process_name );

/* Process */
if (    process_name
&&  strcmp( process_name, "process" ) == 0 )
{
    return 1;
}
return 0;
```

APPLICATION_CLONE (7)

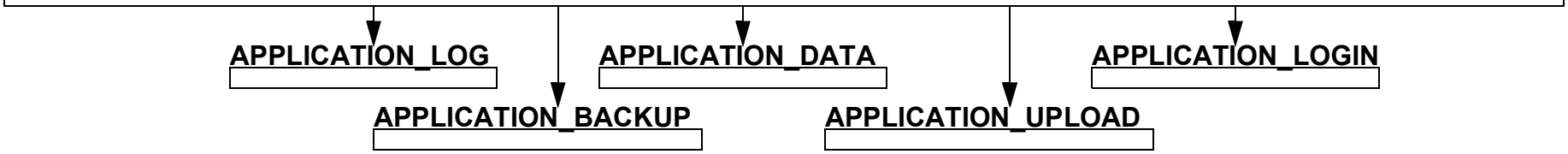
```
/* Driver */
boolean application_clone_post_signup_boolean(
    process_name );

(void)session_fetch(
    current_application_name,
    session_key,
    login_name,
    !application_clone_post_signup_boolean()
        /* appaserver_user_boolean */ );

char *application_clone_sql_gz_filespecification(
    APPLICATION_CLONE_SQL_GZ,
    appaserver_parameter_mount_point() );

APPLICTION_CLONE *application_clone =
    application_clone_new(
        current_application_name,
        login_name,
        security_sql_injection_escape(
            SECURITY_ESCAPE_CHARACTER_STRING,
            destination_application_name ),
        application_clone_sql_gz_filespecification(),
        security_sql_injection_escape(
            SECURITY_ESCAPE_CHARACTER_STRING,
            application_title ),
        application_clone_post_signup_boolean(),
        appaserver_parameter_document_root(),
        appaserver_parameter_data_directory(),
        appaserver_parameter_upload_directory(),
        appaserver_parameter_log_directory(),
        appaserver_parameter_mount_point() );

boolean application_create_execute(
    execute_boolean,
    application_clone->application_create );

if ( !application_create_execute() ) exit( 1 );

if ( application_clone->gz_system_string )
{
    application_clone_system(
        execute_boolean,
        application_clone->gz_system_string );
}
else
{
    application_clone_system(
        execute_boolean,
        application_clone->database_system_string );
}

application_clone_system(
    execute_boolean,
    application_clone->application_system_string );

application_clone_system(
    execute_boolean,
    application_clone->insert_user_system_string );

application_clone_system(
    execute_boolean,
    application_clone->insert_role_system_string );

application_clone_system(
    execute_boolean,
    application_clone->insert_default_system_string );

application_clone_system(
    execute_boolean,
    application_clone->insert_entity_system_string );

application_clone_system(
    execute_boolean,
    application_clone->insert_self_system_string );
```

APPLICATION_CREATE (1)

```
/* Usage */
APPLICATION_CREATE *application_create_new(
    char *application_name,
    char *application_title,
    char *document_root,
    char *data_directory,
    char *upload_directory,
    char *log_directory,
    char *mount_point );

/* Process */
APPLICATION_CREATE *application_create_calloc(
    void );

boolean application_name_invalid(
    SECURITY_ESCAPE_CHARACTER_STRING,
    application_name );

if ( application_name_invalid() ) return this;

char *application_create_database_system_string(
    application_name );
```

```
APPLICATION_LOG *application_log =
    application_log_new(
        APPLICATION_LOG_EXTENSION,
        application_name,
        log_directory );

boolean application_log_exists_boolean(
    application_name,
    application_log->filename );

if ( application_log_exists_boolean() ) return this;

APPLICATION_BACKUP *application_backup =
    application_backup_new(
        APPLICATION_TEMPLATE_NAME,
        application_name,
        mount_point );

APPLICATION_DATA *application_data =
    application_data_new(
        application_name,
        data_directory );
```

```
APPLICATION_UPLOAD *application_upload =
    application_upload_new(
        application_name,
        upload_directory );

APPLICATION_LOGIN *application_login =
    application_login_new(
        APPLICATION_TEMPLATE_NAME,
        APPLICATION_CREATE_DEFAULT_TITLE,
        application_name,
        application_title,
        document_root,
        mount_point );

/* Attributes */
boolean application_name_invalid;
char *database_system_string;
APPLICATION_LOG *application_log;
boolean application_log_exists_boolean;
APPLICATION_BACKUP *application_backup;
APPLICATION_DATA *application_data;
APPLICATION_UPLOAD *application_upload;
APPLICATION_LOGIN *application_login;
```



APPLICATION_CREATE (2)

```
/* Usage */
char *application_create_database_system_string(
    char *application_name );

/* Process */
snprintf(
    system_string[],
    sizeof ( system_string ),
    “echo \”create database %s;\” | sql.e mysql”,
    application_name );

/* Usage */
boolean application_create_system(
    boolean execute_boolean,
    char *system_string );

/* Process */
if ( execute_boolean )
{
    char *spool_fetch(
        system_string,
        1 /* capture_stderr_boolean */ );

    if ( spool_fetch() ) return 0;
}
else
    printf( “<p>%s\n”, system_string );

return 1;

/* Driver */
boolean application_create_execute(
    boolean execute_boolean,
    APPLICATION_CREATE *application_create );

/* Process */
if (    application_create->
            application_name_invalid
||      application_create->
            application_log_exists_boolean )
{
    return 0;
}

boolean application_create_system(
    execute_boolean,
    application_create->database_system_string );

if ( !application_create_system() ) return 0;

(void)application_create_system(
    execute_boolean,
    application_create->
        application_log->
        create_directory_system_string );

(void)application_create_system(
    execute_boolean,
    application_create->
        application_backup->
        create_directory_system_string );

(void)application_create_system(
    execute_boolean,
    application_create->
        application_backup->
        create_file_system_string );

(void)application_create_system(
    execute_boolean,
    application_create->
        application_data->
        create_directory_system_string );

(void)application_create_system(
    execute_boolean,
    application_create->
        application_data->
        texlive_2019_link_system_string );

(void)application_create_system(
    execute_boolean,
    application_create->
        application_data->
        texlive_2021_link_system_string );

(void)application_create_system(
    execute_boolean,
    application_create->
        application_upload->
        create_directory_system_string );

(void)application_create_system(
    execute_boolean,
    application_create->
        application_login->
        directory_system_string );

(void)application_create_system(
    execute_boolean,
    application_create->
        application_login->
        php_file_system_string );

return 1;
```

APPLICATION_LOG

```
/* Usage */
APPLICATION_LOG *application_log_new(
    const char *APPLICATION_LOG_EXTENSION,
    char *application_name,
    char *log_directory );

/* Process */
APPLICATION_LOG *application_log_calloc(
    void );

char *application_log_filename(
    APPLICATION_LOG_EXTENSION,
    application_name,
    log_directory );

char *application_log_create_system_string(
    application_log_filename() );

char *application_log_delete_system_string(
    application_log_filename() );

/* Usage */
char *application_log_filename(
    const char *APPLICATION_LOG_EXTENSION,
    char *application_name,
    char *log_directory );

/* Process */
snprintf(
    filename[],
    sizeof ( filename ),
    "%s/appaserver_%s.%s",
    log_directory,
    application_name,
    APPLICATION_LOG_EXTENSION );

/* Usage */
char *application_log_create_system_string(
    char *application_log_filename() );

/* Process */
snprintf(
    char system_string[],
    sizeof ( system_string ),
    "touch %s && "
    "chmod g+w %s && "
    "chmod o-rwx %s",
    application_log_filename,
    application_log_filename,
    application_log_filename );

/* Usage */
char *application_log_delete_system_string(
    char *application_log_filename() );

/* Process */
snprintf(
    char system_string[],
    sizeof ( system_string ),
    "rm -f %s",
    application_log_filename );

/* Usage */
boolean application_log_exists_boolean(
    char *application_name,
    char *application_log_filename );

/* Process */
if ( string_strcmp( application_name, ... ) == 0 )
        return 1;

boolean application_exists_boolean(
    application_log_filename );

/* Attributes */
char *filename;
char *create_system_string;
char *delete_system_string;
```

APPLICATION_BACKUP

```
/* Usage */
APPLICATION_BACKUP *application_backup_new(
    const char *APPLICATION_TEMPLATE_NAME,
    char *application_name,
    char *mount_point );

/* Process */
APPLICATION_BACKUP *application_backup_calloc(
    void );

char *appaserver_parameter_backup_directory();

static char *application_backup_filename(
    application_name,
    appaserver_parameter_backup_directory() );

char *application_backup_create_file_system_string(
    APPLICATION_TEMPLATE_NAME,
    application_name,
    mount_point,
    application_backup_filename() );

char *application_backup_delete_file_system_string(
    application_backup_filename() );

char *
    application_backup_create_directory_system_string(
        APPLICATION_BACKUP_MKDIR_SCRIPT,
        application_name,
        appaserver_parameter_backup_directory() );

char *application_backup_directory_name(
    application_name,
    appaserver_parameter_backup_directory();

char *
    application_backup_delete_directory_system_string(
        application_backup_directory_name() );

char *application_backup_delete_log_file_system_string(
    char *application_name,
    char *appaserver_parameter_backup_directory() );

/* Usage */
static char *application_backup_filename(
    char *application_name,
    char *appaserver_parameter_backup_directory() );

/* Process */
snprintf(
    static filename[],
    sizeof ( filename ),
    "%s/mysqldump_%s.config",
    appaserver_parameter_backup_directory,
    application_name );

/* Usage */
char *application_backup_create_file_system_string(
    const char *APPLICATION_TEMPLATE_NAME,
    char *application_name,
    char *mount_point,
    char *application_backup_filename() );

/* Process */
snprintf(
    system_string[],
    sizeof ( system_string ),
    "sed 's/change_me/%s/' "
    "< %s/%s/mysqldump_%s.config "
    "> %s && "
    "chmod o-rw %s && "
    "chmod g+w %s",
    application_name,
    mount_point,
    APPLICATION_TEMPLATE_NAME,
    APPLICATION_TEMPLATE_NAME,
    application_backup_filename,
    application_backup_filename,
    application_backup_filename );

/* Usage */
char *application_backup_directory_name(
    char *application_name,
    char *appaserver_parameter_backup_directory );

/* Process */
snprintf(
    static directory_name[],
    sizeof ( directory_name ),
    "%s/%s",
    appaserver_parameter_backup_directory,
    application_name );

/* Usage */
char *
    application_backup_create_directory_system_string(
        const char *APPLICATION_BACKUP_MKDIR_SCRIPT,
        char *application_name,
        char *appaserver_parameter_backup_directory );

/* Process */
snprintf(
    system_string[],
    sizeof ( system_string ),
    "%s %s %s",
    APPLICATION_BACKUP_MKDIR_SCRIPT,
    application_name,
    appaserver_parameter_backup_directory );

/* Usage */
char *application_backup_delete_file_system_string(
    char *application_backup_filename() );

/* Process */
snprintf(
    system_string[],
    sizeof ( system_string ),
    "rm %s",
    application_backup_filename );

/* Usage */
char *
    application_backup_delete_directory_system_string(
        char *application_backup_directory_name() );

/* Process */
snprintf(
    system_string[],
    sizeof ( system_string ),
    "rm -fr %s",
    application_backup_directory );

/* Attributes */
char *appaserver_parameter_backup_directory;
char *filename;
char *create_file_system_string;
char *delete_file_system_string;
char *create_directory_system_string;
char *directory_name;
char *delete_directory_system_string;
char *delete_log_file_system_string;
```

APPLICATION_DATA

```
/* Usage */
APPLICATION_DATA *application_data_new(
    char *application_name,
    char *data_directory );

/* Process */
APPLICATION_DATA *application_data_calloc(
    void );

char *application_data_directory(
    application_name,
    data_directory );

char *application_data_create_directory_system_string(
    application_data_directory() );

char *appaserver_parameter_mount_point();

char *texlive_2019_link_system_string =
    application_data_texlive_link_system_string(
        ".texlive2019",
        application_data_directory(),
        appaserver_parameter_mount_point() );

char *texlive_2021_link_system_string =
    application_data_texlive_link_system_string(
        ".texlive2021",
        application_data_directory(),
        appaserver_parameter_mount_point() );

char *application_data_delete_system_string(
    application_data_directory() );

/* Usage */
char *application_data_directory(
    char *application_name,
    char *data_directory );

/* Process */
snprintf(
    static directory[],
    sizeof ( directory ),
    "%s/%s",
    data_directory,
    application_name );

/* Usage */
char *application_data_create_directory_system_string(
    char *application_data_directory() );

/* Process */
snprintf(
    system_string[],
    sizeof ( system_string ),
    "mkdir %s && "
    "chmod g+ws %s && "
    "chmod o-rwx %s",
    application_data_directory,
    application_data_directory,
    application_data_directory );

/* Usage */
char *application_data_texlive_link_system_string(
    const char *application_data_texlive_directory,
    char *application_data_directory(),
    char *appaserver_parameter_mount_point() );

/* Process */
snprintf(
    system_string[],
    sizeof ( system_string ),
    "ln -s %s/zfonts %s/%s",
    appaserver_parameter_mount_point,
    application_data_directory,
    application_data_texlive_directory );

/* Usage */
char *application_data_delete_system_string(
    char *application_data_directory() );

/* Process */
snprintf(
    system_string[],
    sizeof ( system_string ),
    "rm -r %s",
    application_data_directory );

/* Attributes */
char *directory;
char *create_directory_system_string;
char *texlive_2019_link_system_string;
char *texlive_2021_link_system_string;
char *delete_system_string;
```

APPLICATION_UPLOAD

```
/* Usage */
APPLICATION_UPLOAD *application_upload_new(
    char *application_name,
    char *upload_directory );

/* Process */
APPLICATION_UPLOAD *
    application_upload_calloc(
        void );

char *application_upload_directory(
    application_name,
    upload_directory );

char *
    application_upload_create_directory_system_string(
        application_upload_directory() );

char *application_upload_delete_system_string(
    application_upload_directory() );

/* Usage */
char *application_upload_directory(
    char *application_name,
    char *upload_directory );

/* Process */
snprintf(
    static directory[],
    sizeof ( directory ),
    "%s/%s",
    upload_directory,
    application_name );

/* Usage */
char *
    application_upload_create_directory_system_string(
        char *application_upload_directory() );

/* Process */
snprintf(
    system_string[],
    sizeof ( system_string ),
    "mkdir %s && "
    "chmod g+ws %s && "
    "chmod o-rwx %s",
    application_upload_directory,
    application_upload_directory,
    application_upload_directory );

/* Usage */
char *application_upload_delete_system_string(
    char *application_upload_directory() );

/* Process */
snprintf(
    system_string[],
    sizeof ( system_string ),
    "rm -r %s",
    application_upload_directory );

/* Attributes */
char *directory;
char *create_directory_system_string;
char *delete_system_string;
```

APPLICATION_LOGIN (1)

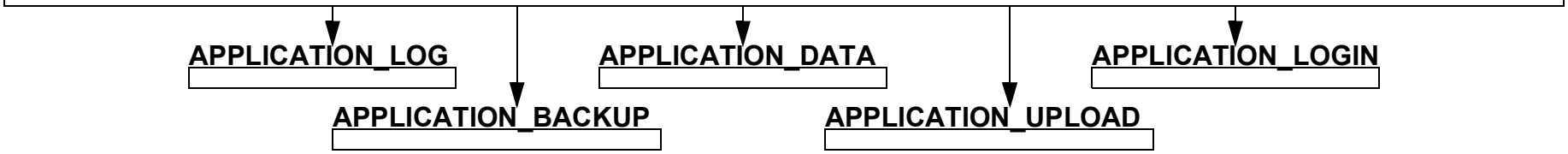
```
/* Usage */
APPLICATION_LOGIN *application_login_new(
    const char *APPLICATION_TEMPLATE_NAME,
    const char *APPLICATION_CREATE_DEFAULT_TITLE,
    char *application_name,
    char *application_title,
    char *document_root,
    char *mount_point );

/* Process */
APPLICATION_LOGIN *application_login_calloc(
    void );

char *application_login_title(
    APPLICATION_CREATE_DEFAULT_TITLE,
    application_title );

char *application_login_destination_directory(
    application_name,
    document_root );

char *application_login_directory_system_string(
    char *application_login_destination_directory() );

char *application_login_source_filename(
    APPLICATION_TEMPLATE_NAME,
    “php” /* extension */,
    mount_point );

char *application_login_destination_filename(
    “php” /* extension */,
    application_login_destination_directory() );

char *php_file_system_string =
    char *application_login_file_system_string(
        application_name,
        application_login_title(),
        application_login_source_filename(),
        application_login_destination_filename() );

char *application_login_delete_system_string(
    application_login_destination_directory() );

/* Attributes */
char *title;
char *destination_directory;
char *directory_system_string;
char *php_file_system_string;
char *delete_system_string;
```

APPLICATION_LOGIN (2)

```
/* Usage */
char *application_login_tile(
    const char *APPLICATION_CREATE_DEFAULT_TITLE,
    char *application_title );

/* Process */
if (    !application_title
||      !*applicationn_title
||      strcmp(
            application_title,
            "application_title" ) == 0 )
{
    return APPLICATION_CREATE_DEFAULT_TITLE;
}
else
{
    return application_title;
}

/* Usage */
char *application_login_destination_directory(
    char *application_name,
    char *document_root );

/* Process */
snprintf(
    static directory[],
    sizeof ( directory ),
    "%s/appaserver/%s",
    document_root,
    application_name );

/* Usage */
char *application_login_source_filename(
    const char *APPLICATION_TEMPLATE_NAME,
    const char *extension,
    char *mount_point );

/* Process */
snprintf(
    static filename[],
    sizeof ( filename ),
    "%s/%s/index.%s",
    mount_point,
    APPLICATION_TEMPLATE_NAME,
    extension );

/* Usage */
char *application_login_destination_filename(
    const char *extension,
    char *application_login_destination_directory() );

/* Process */
snprintf(
    static filename[],
    sizeof ( filename ),
    "%s/index.%s",
    application_login_destination_directory,
    extension );

/* Usage */
char *application_login_file_system_string(
    char *application_name,
    char *application_login_title(),
    char *application_login_source_filename(),
    char *application_login_destination_filename() );

/* Process */
snprintf(
    system_string[],
    sizeof ( system_string ),
    "cat %s | "
    "sed \"s/title_change_me/%s/\" | "
    "sed \"s/application_change_me/%s/\" "
    "> %s && "
    "chmod g+w %s",
    application_login_source_filename,
    application_login_title,
    application_name,
    application_login_destination_filename,
    application_login_destination_filename );

/* Usage */
char *application_login_delete_system_string(
    char *application_login_destination_directory() );

/* Process */
snprintf(
    system_string[],
    sizeof ( system_string ),
    "rm -r %s",
    application_login_destination_directory );
```

APPLICATION_DELETE (1)

```
/* Usage */
APPLICATION_DELETE *application_delete_new(
    char *sql_injection_escape(
        application_name ),
    char *document_root,
    char *data_directory,
    char *upload_directory,
    char *log_directory,
    char *mount_point );

/* Process */
APPLICATION_DELETE *application_delete_calloc(
    void );

boolean application_name_invalid(
    SECURITY_ESCAPE_CHARACTER_STRING,
    application_name );

if ( application_name_invalid() ) return this;

APPLICATION_LOG *application_log =
    application_log_new(
        APPLICATION_LOG_EXTENSION,
        application_name,
        log_directory );

boolean application_log_exists_boolean(
    application_name,
    application_log->filename );

if ( !application_log_exists_boolean() ) return this;

APPLICATION_BACKUP *application_backup =
    application_backup_new(
        APPLICATION_TEMPLATE_NAME,
        application_name,
        mount_point );

APPLICATION_DATA *application_data =
    application_data_new(
        application_name,
        data_directory );

APPLICATION_UPLOAD *application_upload =
    application_upload_new(
        application_name,
        upload_directory );

APPLICATION_LOGIN *application_login =
    application_login_new(
        APPLICATION_TEMPLATE_NAME,
        APPLICATION_CREATE_DEFAULT_TITLE,
        application_name,
        (char *)0 /* application_title */,
        document_root,
        mount_point );

char *application_delete_drop_system_string(
    char *application_name );

/* Attributes */
boolean application_name_invalid;
APPLICATION_LOG *application_log;
boolean application_log_exists_boolean;
APPLICATION_BACKUP *application_backup;
APPLICATION_DATA *application_data;
APPLICATION_UPLOAD *application_upload;
APPLICATION_LOGIN *application_login;
char *drop_system_string;
```



APPLICATION_DELETE (2)

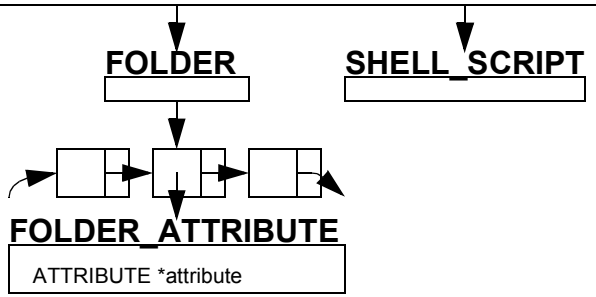
```
/* Usage */
void application_delete_system(
    boolean execute_boolean,
    char *system_string );

/* Process */
if ( execute_boolean )
    system( system_string );
else
    printf( “<p>%s\n”, system_string );

/* Driver */
if ( application_delete->application_name_invalid
||  !application_delete->application_exists_boolean )
{
    exit( 1 );
}

application_delete_system(
    execute_boolean,
    application_delete->
        application_log->
        delete_system_string );

application_delete_system(
    execute_boolean,
    application_delete->
        application_backup->
        delete_file_system_string );

application_delete_system(
    execute_boolean,
    application_delete->
        application_backup->
        delete_directory_system_string );

application_delete_system(
    execute_boolean,
    application_delete->
        application_data->
        delete_system_string );

application_delete_system(
    execute_boolean,
    application_delete->
        application_upload->
        delete_system_string );

application_delete_system(
    execute_boolean,
    application_delete->
        application_login->
        delete_system_string );

application_delete_system(
    execute_boolean,
    application_delete->drop_system_string );

application_delete_system(
    execute_boolean,
    application_delete->
        application_backup->
        delete_log_file_system_string );
```

UPLOAD_PURGE

```
/* Usage */
UPLOAD_PURGE *upload_purge_new(
    char *application_name,
    char *key,
    char *table_name,
    char *column_name,
    char *appaserver_parameter_upload_directory() );

/* Process */
UPLOAD_PURGE *upload_purge_calloc(
    void );

char *upload_purge_directory(
    char *application_name,
    char *key,
    char *appaserver_parameter_upload_directory );

LIST *application_not_null_data_list(
     table_name,
     column_name );

char *upload_purge_filename_system_string(
    UPLOAD_PURGE_OLDER_MINUTES,
    upload_purge_directory() );

LIST *upload_purge_filename_list(
    char *upload_purge_filename_system_string() );

LIST *upload_purge_system_list(
    application_not_null_data_list(),
    upload_purge_filename_list() );

/* Usage */
char *upload_purge_filename_system_string(
    const int UPLOAD_PURGE_OLDER_MINUTES,
    char *upload_purge_directory() );

/* Process */
snprintf(
    static system_string[],
    sizeof ( system_string ),
    "cd %s && "
    "find . -type f -mmin +%d -print | "
    "sed 's,^./,,'",
    upload_purge_directory,
    UPLOAD_PURGE_OLDER_MINUTES );

/* Usage */
LIST *upload_purge_system_list(
    LIST *appliction_not_null_data_list(),
    LIST *upload_purge_filename_list() );

/* Process */
LIST *system_list = list_new();

for char *filename in upload_purge_filename_list
{
    boolean list_string_exists(
        filename,
        application_not_null_data_list );

if ( !list_string_exists() )
{
    char *upload_purge_system_string(
        char *filename );

    list_set(
        system_list,
        upload_purge_system_string() );
}

/* Driver */
void upload_purge_execute(
    char *upload_purge->directory,
    LIST *upload_purge->system_list,
    boolean display_only_boolean );

/* Process */
chdir( directory );

for char *system_string in system_list
{
    if ( display_only_boolean )
        printf( "%s\n", system_string );
    else
        system( system_string );
}

/* Attributes */
char *directory;
LIST *application_not_null_data_list;
char *filename_system_string;
LIST *filename_list;
LIST *system_list;
```

CREATE_TABLE (1)

```
/* Usage */
CREATE_TABLE *create_table_new(
    char *application_name,
    char *sql_injection_escape(
        folder_name ),
    boolean execute_boolean,
    char *appaserver_parameter->data_directory );

/* Process */
CREATE_TABLE *create_table_calloc(
    void );

FOLDER *folder =
    folder_fetch(
        folder_name,
        (LIST *)0 /* exclude_attribute_name_list */,
        /* Sets folder_attribute_primary_list */
        /* Sets folder_attribute_primary_key_list */
        /* Sets folder_attribute_name_list */
        1 /* fetch_folder_attribute_list */,
        1 /* fetch_attribute */,
        0 /* not cache_boolean */ );

char *appaserver_table_name(
    folder_name );

char *create_table_drop_statement(
    folder->create_view_statement,
    appaserver_table_name() );

char *create_table_statement(
    folder->create_view_statement,
    folder->folder_attribute_list,
    folder->data_directory,
    folder->index_directory,
    folder->storage_engine,
    appaserver_table_name() );

char *create_table_unique_index_statement(
    CREATE_TABLE_UNIQUE_SUFFIX,
    folder->folder_attribute_primary_key_list,
    appaserver_table_name() );

char *create_table_additional_unique_statement(
    CREATE_TABLE_ADDITIONAL_SUFFIX,
    folder->folder_attribute_list,
    appaserver_table_name() );

LIST *create_table_additional_index_list(
    folder->folder_attribute_list,
    appaserver_table_name() );

char *shell_script_filespecification(
    "create_table" /* PROCESS_LABEL */,
    char *application_name,
    char *folder_name,
    char *data_directory );

LIST *create_table_sql_statement_list(
    char *create_table_drop_statement(),
    char *create_table_statement(),
    char *create_table_unique_index_statement(),
    char *create_table_additional_unique_statement(),
    LIST *create_table_additional_index_list() );

SHELL_SCRIPT *create_table_shell_script(
    application_name,
    execute_boolean,
    shell_script_filespecification(),
    create_table_sql_statement_list() );

/* Driver */
if ( execute_boolean )
{
    system( create_table->shell_script_filespecification );

    if ( *process_name )
    {
        process_increment_count( process_name );
        printf( "<h3>Process complete.<h3>\n" );
    }
}
else
{
    list_html_table_display(
        create_table->sql_statement_list,
        "SQL Statement" /* heading_string */ );

    printf(
        "<h3>Created: %s</h3>\n",
        create_table->shell_script_filespecification );
}

/* Attributes */
FOLDER *folder;
char *appaserver_table_name;
char *drop_statement;
char *statement;
char *unique_index_statement;
LIST *additional_unique_index_list;
LIST *additional_index_list;
char *shell_script_filespecification;
LIST *sql_statement_list;
SHELL_SCRIPT *shell_script;
```



CREATE_TABLE (2)

```
/* Usage */
char *create_table_drop_statement(
    char *create_view_statement,
    char *appaserver_table_name );

/* Process */
if ( create_view_statement )
{
    snprintf(
        static statement[],
        sizeof ( statement ),
        "drop view if exists %s;",
        appaserver_table_name );
}
else
{
    snprintf(
        static statement[],
        sizeof ( statement ),
        "drop table if exists %s;",
        appaserver_table_name );
}

/* Usage */
char *create_table_statement(
    char *create_view_statement,
    LIST *folder->folder_attribute_list,
    char *data_directory,
    char *index_directory,
    char *storage_engine,
    char *appaserver_table_name );

/* Process */
char statement[];
char *ptr = statement;
boolean first_time = 1;

if ( create_view_statement )
{
    return create_view_statement;
}

ptr += sprintf( ptr,
    "create table %s (",
    appaserver_table_name );

for FOLDER_ATTRIBUTE *folder_attribute in
    folder_attribute_list
{
    if ( first_time )
        first_time = 0;
    else
        ptr += sprintf( ptr, "," );

    char *attribute_database_datatype(
        folder_attribute->attribute->datatype_name,
        folder_attribute->attribute->width,
        folder_attribute->attribute->float_decimal_places,
        folder_attribute->primary_key_index,
        folder_attribute->default_value );

    ptr += sprintf( ptr,
        "%s %s",
        folder_attribute->attribute_name,
        attribute_database_datatype() );
}

ptr += sprintf( ptr, ")" );

if ( data_directory )
{
    ptr += sprintf( ptr,
        " data directory = '%s',",
        data_directory );
}

if ( index_directory )
{
    ptr += sprintf( ptr,
        " index directory = '%s',",
        index_directory );
}

char *create_table_storage_engine(
    const char *CREATE_TABLE_MYISAM_ENGINE,
    char *storage_engine );

ptr += sprintf( ptr,
    " engine %s;",
    create_table_storage_engine() );

/* Usage */
char *create_table_unique_index_statement(
    const char *CREATE_TABLE_UNIQUE_SUFFIX,
    LIST *folder->folder_attribute_primary_key_list,
    char *appaserver_table_name );

/* Process */
char *create_table_unique_index_name(
    CREATE_TABLE_UNIQUE_SUFFIX,
    appaserver_table_name );

snprintf(
    static statement[],
    sizeof ( statement ),
    "create unique index %s on %s (%s);"
    create_table_unique_index_name(),
    appaserver_table_name,
    list_display_delimited(
        folder_attribute_primary_key_list,
        ',' ) );
```

CREATE_TABLE (3)

```
/* Usage */
char *create_table_additional_unique_statement(
    const char *CREATE_TABLE_ADDITIONAL_SUFFIX,
    LIST *folder->folder_attribute_list,
    char *appaserver_table_name );

/* Process */
char statement[];
char *ptr = statement;

LIST *create_table_additional_unique_name_list(
    LIST *folder_attribute_list );

if ( !list_length( create_table_additional_unique_name_list() )
        return NULL;

char *create_table_additional_unique_index_name(
    const char *CREATE_TABLE_ADDITIONAL_SUFFIX,
    char *appaserver_table_name );

ptr += sprintf( ptr,
    "create unique index %s on %s (",
    create_table_additional_unique_index_name(),
    appaserver_table_name );

for char *attribute_name in
    create_table_additional_unique_name_list()
{
    if ( !list_at_first(
            create_table_additional_unique_name_list() )
    {
        ptr += sprintf( ptr, "," );
    }

    ptr += sprintf( ptr,
        "%s",
        attribute_name );
}

ptr += sprintf( ptr, ");" );
```

```
/* Usage */
char *create_table_unique_index_name(
    char *CREATE_TABLE_UNIQUE_SUFFIX,
    char *appaserver_table_name );

/* Process */
if ( CREATE_TABLE_UNIQUE_SUFFIX )
{
    snprintf(
        static unique_name[],
        sizeof ( unique_name ),
        "%s_%s",
        appaserver_table_name,
        CREATE_TABLE_UNIQUE_SUFFIX );
}
else
{
    string_strcpy(
        static unique_name[],
        appaserver_table_name,
        sizeof ( unique_name ) /* buffer_size */ );
}

/* Usage */
LIST *create_table_additional_index_list(
    LIST *folder->folder_attribute_list,
    char *appaserver_table_name );

/* Process */
LIST *create_table_additional_index_name_list(
    LIST *folder_attribute_list );

for char *additional_index_name in
    create_table_additional_index_name_list()
{
    snprintf(
        statement[],
        sizeof ( statement ),
        "create index %s on %s (%s);",
        additional_index_name,
        appaserver_table_name,
        additional_index_name );

    list_set(
        list,
        strdup( statement ) );
}

/* Usage */
SHELL_SCRIPT *create_table_shell_script(
    char *application_name,
    boolean execute_boolean,
    char *shell_script_filespecification,
    LIST *create_table_sql_statement_list );

/* Process */
SHELL_SCRIPT *shell_script_new(
    application_name,
    shell_script_filespecification,
    create_table_sql_statement_list,
    execute_boolean /* tee_appaserver_boolean */,
    execute_boolean /* html_paragraph_..._boolean */ );

/* Usage */
char *create_table_primary_index_system_string(
    CREATE_TABLE_UNIQUE_SUFFIX,
    LIST *primary_key_list,
    char *appaserver_table_name );

/* Process */
char *create_table_unique_index_statement(
    CREATE_TABLE_UNIQUE_SUFFIX,
    primary_key_list,
    appaserver_table_name );

snprintf(
    static system_string[],
    sizeof ( system_string ),
    "echo \"%s\" | "
    "tee_appaserver.sh | "
    "sql.e 2>&1",
    create_table_unique_index_statement() );
```

DATE_CONVERT (1)

```
/* Usage */
DATE_CONVERT *date_convert_new(
    enum date_convert_format_enum
        source_enum,
    enum date_convert_format_enum
        destination_enum,
    char *source_date_string );

/* Process */
DATE_CONVERT *date_convert_calloc( void );

if ( !source_date_string || !*source_date_string )
{
    this->source_enum = date_convert_unknown;
    return this;
}

int strlen( source_date_string );

if ( strlen() > 20 )
{
    this->source_enum = date_convert_unknown;
    return this;
}

int string_character_count(
    ' ',
    source_date_string );

if ( string_character_count() > 1 )
{
    this->source_enum = date_convert_unknown;
    return this;
}

column(
    char date_column_string[],
    0,
    source_date_string );

if ( string_character_count() == 1 )
{
    column(
        char time_column_string[],
        1,
        source_date_string );
}

if ( source_enum == date_convert_unknown )
{
    this->source_enum =
        date_convert_string_evaluate(
            date_column_string
                /* source_date_string */ );

    if ( this->source_enum == date_convert_unknown )
        return this;
}

char *return_date_string =
    date_convert_return_date_string(
        this->source_enum,
        destination_enum,
        date_column_string
            /* source_date_string */ );

if ( !return_date_string )
{
    this->source_enum =
        date_convert_string_evaluate(
            date_column_string
                /* source_date_string */ );

    if ( this->source_enum == date_convert_unknown )
    {
        return this;
    }

    char *return_date_string =
        date_convert_return_date_string(
            this->source_enum,
            destination_enum,
            date_column_string
                /* source_date_string */ );
}

if ( !return_date_string ) exit( 1 );

if ( *time_column_string )
{
    sprintf(char tmp_string[],
            "%s %s",
            return_date_string,
            time_column_string );

    if ( return_date_string != date_column_string )
    {
        free( return_date_string );
    }

    return_date_string = strdup( tmp_string );
}

/* Usage */
char *date_convert_return_date_string(
    enum date_convert_format_enum
        source_enum,
    enum date_convert_format_enum
        destination_enum,
    char *source_date_string );

/* Process */
if ( source_enum == date_convert_american )
{
    char *return_date_string =
        char *date_convert_source_american(
            destination_enum,
            source_date_string );
}
else
if ( source_enum == date_convert_international )
{
    char *return_date_string =
        char *date_convert_source_international(
            destination_enum,
            source_date_string );
}
else
if ( source_enum == date_convert_military )
{
    char *return_date_string =
        char *date_convert_source_military(
            destination_enum,
            source_date_string );
}
else
{
    exit( 1 );
}
```

DATE_CONVERT (2)

```
/* Usage */
char *date_convert_source_american(
    enum date_convert_format_enum
        destination_enum,
    char *source_date_string );

/* Process */
if ( destination_enum == date_convert_military )
{
    char *date_convert_american_to_international(
        source_date_string );

    char *date_convert_international_to_military(
        date_convert_american_to_international() );
}
else
if ( destination_enum == date_convert_international )
{
    char *date_convert_american_to_international(
        source_date_string );
}
else
{
    boolean date_convert_is_valid_american(
        source_date_string );

    if ( !date_convert_is_valid_american() )
        return (char *)0;
    else
        return source_date_string;
}

/* Usage */
char *date_convert_source_international(
    enum date_convert_format_enum
        destination_enum,
    char *source_date_string );

/* Process */
if ( destination_enum == date_convert_military )
{
    char *date_convert_international_to_military(
        source_date_string );
}
else
if ( destination_enum == date_convert_american )
{
    char *date_convert_international_to_american(
        source_date_string );
}
else
{
    boolean date_convert_is_valid_international(
        source_date_string );

    if ( !date_convert_is_valid_international() )
        return (char *)0;
    else
        return source_date_string;
}

/* Usage */
char *date_convert_source_military(
    enum date_convert_format_enum
        destination_enum,
    char *source_date_string );

/* Process */
if ( destination_enum == date_convert_international )
{
    char *date_convert_military_to_international(
        source_date_string );
}
else
if ( destination_enum == date_convert_american )
{
    char *date_convert_military_to_international(
        source_date_string );
    char *date_convert_international_to_american(
        date_convert_military_to_international()
            /* source_date_string */ );
}
else
{
    boolean date_convert_is_valid_military(
        source_date_string );

    if ( !date_convert_is_valid_military() )
        return (char *)0;
    else
        return source_date_string;
}

/* Usage */
char *date_convert_international_to_military(
    char *source_date_string );

/* Process */

/* Usage */
char *date_convert_american_to_international(
    char *source_date_string );

/* Process */

/* Usage */
char *date_convert_military_to_international(
    char *source_date_string );

/* Process */

/* Usage */
char *date_convert_international_to_american(
    char *source_date_string );

/* Process */
```

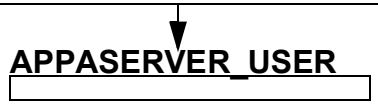
```
/* Usage */
enum date_convert_format_enum
   date_convert_string_evaluate(
      char *source_date_string );

/* Process */
column( char date_string[], 0, source_date_string );

boolean date_convert_is_valid_military(
   char *date_string );

boolean date_convert_is_valid_international(
   char *date_string );

boolean date_convert_is_valid_american(
   char *date_string );

/* Usage */
static enum date_convert_format_enum
   date_convert_login_name_enum(
      char *application_name,
      char *login_name );

/* Process */
static enum date_convert_format_enum
   format_enum =
      date_convert_unknown;

if ( format_enum != date_convert_unknown )
{
   return format_enum;
}

APPASERVER_USER *appaserver_user =
   appaserver_user_login_fetch(
      login_name,
      0 /* not fetch_role_name_list */ );

if (    appaserver_user->user_date_format
&&   *appaserver_user->user_date_format )
{
   format_enum =
      date_convert_format_string_evaluate(
         appaserver_user->user_date_format );

   return format_enum;
}

APPLICATION *application =
   application_fetch(
      application_name );

if (    application->user_date_format
&&   *application->user_date_format )
{
   format_enum =
      date_convert_format_string_evaluate(
         application->user_date_format );
}
else
{
   format_enum = date_convert_blank;
}

/* Usage */
DATE_CONVERT *
   date_convert_destination_international(
      char *application_name,
      char *login_name,
      char *date_string );

/* Process */
static enum date_convert_format_enum
   source_enum = date_convert_blank;

if ( source_enum == date_convert_blank )
{
   enum date_convert_format_enum source_enum =
      date_convert_login_name_enum(
         application_name,
         login_name );
}

DATE_CONVERT *date_convert =
   date_convert_new(
      source_enum,
      date_convert_international /* destination_enum */,
      date_string );

/* Public */
enum date_convert_format_enum
   date_convert_format_string_evaluate(
      char *date_convert_format_string );

boolean date_convert_is_valid_integers(
   int year_integer,
   int month_integer,
   int day_integer );

char *date_convert_american_sans_slashes(
   char *compressed_date_string );

char *date_convert_select_format_string(
   enum date_convert_format_enum
      date_convert_format_enum );

char *date_convert_format_string(
   enum date_convert_format_enum
      date_convert_format_enum );

void date_convert_free(
   DATE_CONVERT *date_convert );

/* Attributes */
enum date_convert_format_enum source_enum;
enum date_convert_format_enum destination_enum;
char *source_date_string;
char date_column_string[];
char time_column_string[];
char *return_date_string;
```

enum date_convert_format_enum

date_convert_international,
date_convert_american,
date_convert_military,
date_convert_unknown
date_convert_blank

SESSION (1)

```
/* Usage */
SESSION *session_new(
    char *application_name,
    char *login_name,
    boolean appaserver_user_boolean );

/* Process */
SESSION *session_calloc(
    void );

char *session_key( application_name );
char *session_login_date();
char *session_login_time();

char *environment_http_user_agent(
    ENVIRONMENT_HTTP_USER_AGENT_KEY ),

char *session_http_user_agent(
    SESSION_USER_AGENT_WIDTH,
    environment_http_user_agent );

char *environment_remote_ip_address(
    ENVIRONMENT_REMOTE_ADDR_KEY ),

char *remote_ip_address =
    char *session_current_ip_address(
        char *environment_remote_ip_address() );

if ( appaserver_user_boolean )
{
    APPASERVER_USER *appaserver_user_login_fetch(
        login_name,
        0 /* not fetch_role_name_list */ );

    if ( !appaserver_user_login_fetch() ) exit( 1 );
}

boolean entity_contact_key_boolean(
    ENTITY_TABLE,
    ENTITY_CONTACT_KEY_COLUMN );

char *session_insert_string(
    SQL_DELIMITER,
    session_key(),
    session_login_date(),
    session_login_time(),
    session_http_user_agent(),
    session->remote_ip_address,
    session->appaserver_user,
    entity_contact_key_boolean() );

char *session_insert_system_string(
    SQL_DELIMITER,
    SESSION_TABLE,
    SESSION_INSERT,
    entity_contact_key_boolean() );

/* Attributes */
char *session_key;
char *login_date;
char *login_time;
char *remote_ip_address;
char *http_user_agent;
APPASERVER_USER *appaserver_user;
boolean entity_contact_key_boolean;
char *insert_string;
char *insert_system_string;

/* Fetch attributes */
char *current_ip_address;
boolean remote_ip_address_changed_boolean;
char *last_access_date;
char *last_access_time;
char *update_sql;
char *purge_temporary_files_system_string;
```



SESSION (2)

```
/* If error then exit( 1 ) */
SESSION *session_fetch(
    char *application_name,
    char *session_key,
    char *login_name,
    boolean appaserver_user_boolean );

/* Process */
boolean entity_contact_key_boolean(
    ENTITY_TABLE,
    ENTITY_CONTACT_KEY_COLUMN );

char *entity_select_string(
    SESSION_SELECT /* ENTITY_SELECT */,
    ENTITY_CONTACT_KEY_COLUMN,
    entity_contact_key_boolean() );

char *session_primary_where(
    char *session_key );

char *appaserver_system_string(
    entity_select_string(),
    SESSION_TABLE,
    session_primary_where() );

char *string_fetch( appaserver_system_string() );

SESSION *session_parse(
    appaserver_user_boolean,
    entity_contact_key_boolean(),
    string_fetch() );

char *environment_remote_ip_address(
    ENVIRONMENT_REMOTE_ADDR_KEY ) );

char *session_current_ip_address(
    environment_remote_ip_address() );

if ( !session_parse() )
{
    void session_access_failed_message_exit(
        application_name,
        login_name,
        session_current_ip_address() );
}

boolean
    appaserver_parameter_bypass_ip_changed_check(
        application_name );

if ( !appaserver_parameter_bypass_ip_changed_check() )
{
  boolean session_remote_ip_address_changed_boolean(
    session_parse()->remote_ip_address,
    session_currrent_ip_address() );
}

if ( session_remote_ip_address_changed_boolean() )
{
    void session_message_ip_address_changed_exit(
        application_name,
        session_key,
        login_name,
        session_parse()->remote_ip_address,
        session_current_ip_address() );
}

boolean session_browser_key_changed_boolean(
    session_parse()->http_user_agent );

if ( session_browser_key_changed_boolean() )
{
    void session_access_failed_message_exit(
        application_name,
        login_name,
        session_current_ip_address() );
}

char *session_last_access_date();
char *session_last_access_time();

char *session_update_sql(
    SESSION_TABLE,
    session_key,
    session_last_access_date(),
    session_last_access_time() );

(void)sql_execute(
    SQL_EXECUTABLE,
    (char *)0 /* appaserver_error_filename */,
    (LIST *)0 /* sql_list */,
    session_update_sql() );

char *session_purge_temporary_files_system_string(
    application_name );

system(
    session_purge_temporary_files_system_string() );
```

```
/* Usage */
SESSION *session_parse(
    boolean appaserver_user_boolean,
    boolean entity_contact_key_boolean,
    char *string_fetch );

/* Process */
SESSION *session_calloc();

if ( appaserver_user_boolean )
{
    if ( entity_contact_key_boolean )
    {
        piece( contact_key[], SQL_DELIMITER, string_fetch, 8 );

    APPASERVER_USER *appaserver_user_fetch(
        full_name[] /* stack memory */,
        contact_key[] /* stack memory */,
        entity_contact_key_boolean,
        0 /* not fetch_role_name_list */ );

    if ( !appaserver_user_fetch() ) return NULL;
}

/* Usage */
char *session_http_user_agent(
    const int SESSION_USER_AGENT_WIDTH
        /* probably 80 */,
    char *environment_http_user_agent );

/* Process */
char *string_left(
    user_agent[] /* destination */,
    environment_http_user_agent,
    SESSION_USER_AGENT_WIDTH /* how_many */ );

/* Usage */
char *session_login_date(
    void );

/* Process */
char *date_now_yyyy_mm_dd(
    date_utc_offset() );

/* Usage */
char *session_login_time(
    void );

/* Process */
char *date_now_hhmm(
    date_utc_offset() );

/* Usage */
char *session_last_access_date(
    void );

/* Process */
char *session_login_date();

/* Usage */
char *session_last_access_time(
    void );

/* Process */
char *session_login_time();

/* Usage */
char *session_insert_string(
    const char SQL_DELIMITER,
    char *session_key,
    char *login_date,
    char *login_time,
    char *http_user_agent,
    char *remote_ip_address,
    APPASERVER_USER *appaserver_user,
    boolean entity_contact_key_boolean );

/* Process */
char *session_insert_appaserver_user_string(
    SQL_DELIMITER,
    appaserver_user,
    entity_contact_key_boolean );

snprintf(
    insert_string[],
    sizeof ( insert_string ),
    "%s^%s^%s^%s^%s^%s^%s^%s\n",
    session_key,
    login_date,
    login_time,
    login_date,
    login_time,
    http_user_agent,
    remote_ip_address,
    session_insert_appaserver_user_string() );

/* Usage */
char *session_insert_system_string(
    const char SQL_DELIMITER,
    const char *SESSION_TABLE,
    const char *SESSION_INSERT,
    boolean entity_contact_key_boolean );

/* Process */
char *entity_insert_column_string(
    SESSION_INSERT /* ENTITY_INSERT */,
    APPASERVER_USER_CONTACT_KEY_COLUMN,
    entity_contact_key_boolean );

snprintf(
    system_string[],
    sizeof ( system_string ),
    "insert_statement.e table=%s field=\"%s\" delimiter='%c' |"
    "tee_appaserver.sh |"
    "sql.e",
    SESSION_TABLE,
    entity_insert_column_string(),
    SQL_DELIMITER );

/* Usage */
char *session_insert_appaserver_user_string(
    const char SQL_DELIMITER,
    APPASERVER_USER *appaserver_user,
    boolean entity_contact_key_boolean );

/* Process */
if ( !appaserver_user )
{
    if ( entity_contact_key_boolean )
            char *appaserver_user_string = "^";
    else
            char *appaserver_user_string = "";
}
else
{
    OPTIONAL_COLUMN *optional_column_new(
        SQL_DELIMITER,
        appaserver_user->full_name /* base_string */,
        appaserver_user->contact_key /* component */,
        0 /* not escape_boolean */,
        entity_contact_key_boolean /* set_boolean */ );

    char *appaserver_user_string =
        heap char *optional_column_new()->return_string;
}
```

```
/* Usage */
char *session_purge_temporary_files_system_string(
    char *application_name );

/* Process */
snprintf(
    static system_string[],
    sizeof ( system_string ),
    "appaserver_purge_temporary_files.sh %s &",
    application_name );

/* Usage */
void session_insert(
    char *session_insert_string,
    char *session_insert_system_string );

/* Process */
FILE *appaserver_output_pipe(
    session_insert_system_string );

fprintf(
    appaserver_output_pipe(),
    "%s\n",
    session_insert_string );

pclose( appaserver_output_pipe() );

/* Usage */
char *session_key(
    char *application_name );

/* Process */
snprintf(
    system_string[],
    sizeof ( system_string ),
    "session_number_new.sh %s",
    application_name );

char *string_pipe_fetch( system_string );

/* Usage */
boolean session_sql_injection_strcmp_okay(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name );

/* Process */

/* Usage */
void session_sql_injection_message_exit(
    char *argv_0,
    char *environment_remote_ip_address() );

/* Process */

/* Usage */
void session_access_failed_message_exit(
    char *application_name,
    char *login_name,
    char *session_current_ip_address() );

/* Process */

/* Usage */
void session_environment_set(
    char *application_name );

/* Process */
void environment_session_set( application_name );

/* Usage */
static char *session_update_sql(
    const char *SESSION_TABLE,
    char *session_key,
    char *session_last_access_date,
    char *session_last_access_time );

/* Process */

/* Usage */
boolean session_remote_ip_address_changed_boolean(
    char *session_parse()->remote_ip_address,
    char *session_currrent_ip_address );

/* Process */
```

SESSION (5)

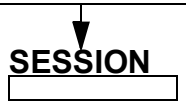
```
/* Usage */
void session_message_ip_address_changed_exit(
    char *application_name,
    char *session_key,
    char *login_name,
    char *session_parse()->remote_ip_address,
    char *session_current_ip_address );

/* Process */

/* Usage */
void session_delete(
    const char *SESSION_TABLE,
    char *application_name,
    char *session->session_key );

/* Process */
char *appaserver_error_filespecification(
    application_name );

static char *session_delete_sql(
    const char *SESSION_TABLE,
    char *session_key );

(void)sql_execute(
    SQL_EXECUTABLE,
    appaserver_error_filespecification(),
    (LIST *)0 /* sql_list */,
    session_delete_sql() );

/* Usage */
char *session_right_trim(
    char *directory_filename_session_suffix );

/* Process */

/* Usage */
void session_folder_permission_message_exit(
    char *application_name,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *state1,
    char *session_current_ip_address );

/* Process */

/* Usage */
boolean session_browser_key_changed_boolean(
    char *session_parse()->http_user_agent );

/* Process */
char *environment_http_user_agent(
    ENVIRONMENT_HTTP_USER_AGENT_KEY ),

char *session_http_user_agent(
    SESSION_USER_AGENT_WIDTH,
    environment_http_user_agent );

int strcmp(
    http_user_agent,
    session_http_user_agent() );

if ( strcmp() != 0 )
        boolean changed_boolean = 1;
else
        boolean changed_boolean = 0;
```

SESSION_FOLDER

```
/* Usage */
/* Sets appaserver environment and outputs argv. */
/* Each parameter is security inspected. */
/* Any error will exit( 1 ). */
SESSION_FOLDER *session_folder_integrity_exit(
    int argc,
    char **argv,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *state1,
    char *state2 );

/* Process */
SESSION_FOLDER *session_folder_calloc(
    void );

char *application_name =
    security_sql_injection_escape(
            SECRUITY_ESCAPE_CHARACTER_STRING,
        application_name );

session_environment_set( application_name );

appaserver_error_argv_append_file(
    argc, argv, application_name );

char *session_key =
    security_sql_injection_escape(
            SECRUITY_ESCAPE_CHARACTER_STRING,
        session_key );

char *login_name =
    security_sql_injection_escape(
            SECRUITY_ESCAPE_CHARACTER_STRING,
        login_name );

char *role_name =
    security_sql_injection_escape(
            SECRUITY_ESCAPE_CHARACTER_STRING,
        role_name );

char *folder_name =
    security_sql_injection_escape(
            SECRUITY_ESCAPE_CHARACTER_STRING,
        folder_name );

char *state1 =
    security_sql_injection_escape(
            SECRUITY_ESCAPE_CHARACTER_STRING,
        state1 );

char *session_folder_state2(
    APPASERVER_LOOKUP_STATE,
    APPASERVER_UPDATE_STATE,
    state1,
    state2 );

SESSION *session =
    /* If error then exit( 1 ) */
    session_fetch(
        application_name,
        session_key,
        login_name,
        1 /* appaserver_user_boolean */ );

LIST *role_folder_list(
    role_name,
    folder_name );

boolean session_folder_valid(
    char *state1,
    char *session_folder_state2(),
    char *folder_name,
    LIST *role_folder_list() );

if (!session_folder_valid() )
{
    session_folder_permission_message_exit(
        application_name,
        login_name,
        role_name,
        folder_name,
        state,
        session->current_ip_address );
}

/* Usage */
char *session_folder_state2(
    const char *APPASERVER_LOOKUP_STATE,
    const char *APPASERVER_UPDATE_STATE,
    char *state1,
    char *state2 );

/* Process */
if ( state2 )
{
    session_folder_state2 = state2;
}
else
if (    strcmp(
            state1,
            APPASERVER_LOOKUP_STATE ) == 0 )
{
    session_folder_state2 =
        APPASERVER_UPDATE_STATE;
}

/* Attributes */
char *application_name;
char *session_key;
char *login_name;
char *role_name;
char *folder_name;
char *state1;
char *state2;
LIST *role_folder_list;
SESSION *session;
```

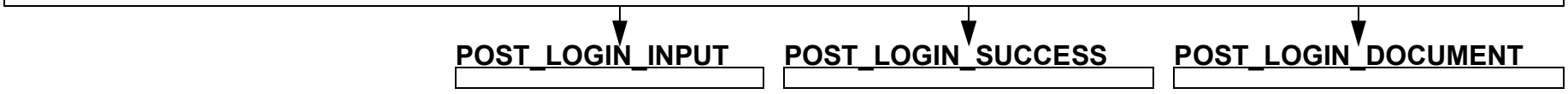
SESSION_PROCESS

```
/* Usage */
/* Sets appaserver environment and outputs argv. */
/* Each parameter is security inspected. */
/* Any error will exit( 1 ). */
SESSION_PROCESS *session_process_integrity_exit(
    int argc,
    char **argv,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *process_name );

/* Process */
if ( !session_sql_injection_strcmp_okay(
    application_name,
    session_key,
    login_name,
    role_name ) )
{
    session_sql_injection_message_exit(
        argv[ 0 ],
        environment_remote_ip_address(
            ENVIRONMENT_REMOTE_ADDR_KEY ) );
}

session_environment_set( application_name );

appaserver_error_argv_append_file(
    argc, argv, application_name );

SESSION_PROCESS *session_process_calloc(
    void );

SESSION *session =
    /* If error then exit( 1 ) */
    session_fetch(
        application_name,
        session_key,
        login_name,
        1 /* appaserver_user_boolean */ );

char *role_name =
    security_sql_injection_escape(
        SECRUITY_ESCAPE_CHARACTER_STRING,
        role_name );

boolean session_process_valid(
    char *security_sql_injection_escape(
        SECRUITY_ESCAPE_CHARACTER_STRING,
        process_name ),
    LIST *role_process_list(
        security_sql_injection_escape_role_name,
        0 /* not fetch_process */ ),

if ( !session_process_valid() )
{
    session_access_failed_message_exit(
        application_name,
        login_name,
        session_current_ip_address() );
}

/* Attributes */
char *role_name;
char *process_name;
char *process_set_name;
SESSION *session;
```



POST_LOGIN (1)

```
/* Usage */
POST_LOGIN *post_login_new(
    int argc,
    char **argv );

/* Process */
POST_LOGIN *post_login_calloc(
    void );

POST_LOGIN_INPUT *
    post_login_input =
        post_login_input_new(
        argc,
        argv );

if (    !post_login_input->missing_application_boolean
&&  !post_login_input->public_boolean
&&  !post_login_input->
          form_password_blank_boolean
&&  !post_login_input->missing_login_name_boolean
&&  !post_login_input->user_not_found_boolean )
{
    char *post_login_security_encrypt_password(
            post_login_input->form_password,
            post_login_input->
               appaserver_user->
               password );
}

if (    !post_login_input->missing_application_boolean
&&  !post_login_input->public_boolean
&&  !post_login_input->user_not_found_boolean
&&  !post_login_input->missing_login_name_boolean )
{
    boolean post_login_password_fail_boolean(
        boolean post_login_input->
            email_address_boolean,
        char *post_login_input->
            appaserver_user->
            password,
        boolean post_login_input->
            form_password_blank_boolean,
        char *post_login_security_encrypt_password() );
}

char *post_login_reject_index_html_parameter(
    boolean post_login_input->
        missing_application_boolean,
    boolean post_login_input->
        invalid_application_boolean,
    boolean post_login_input->
        missing_login_name_boolean,
    boolean post_login_input->
        invalid_login_name_boolean,
    boolean !post_login_input->
        application_exists_boolean
          /* application_not_exists_boolean */,
    boolean post_login_input->
        user_not_found_boolean,
    boolean post_login_input->
        deactivated_boolean,
    boolean post_login_password_fail_boolean() );

if ( post_login_reject_index_html_parameter() )
{
    POST_LOGIN_DOCUMENT *
        post_login_document =
            post_login_document_new(
                post_login_input->dictionary,
                post_login_input->application_name,
                post_login_reject_index_html_parameter() );

    return this;
}

POST_LOGIN_SUCCESS *post_login_success =
    post_login_success_new(
        post_login_input->application_name,
        post_login_input->login_name,
        post_login_input->password_blank_boolean,
        post_login_input->email_address_boolean );

if (    post_login_input->email_address_boolean
&&  post_login_input->form_password_blank_boolean )
{
    POST_LOGIN_DOCUMENT *post_login_document =
        post_login_document_new(
            post_login_input->dictionary,
            post_login_input->application_name,
            post_login_success->index_html_parameter );
}

/* Usage */
char *post_login_security_encrypt_password(
    char *post_login_input_form_password,
    char *appaserver_user_password );

/* Process */
if ( !post_login_input_form_password ) return NULL;

/* If appaserver_user_password isn't encrypted */
if ( string_strcmp(
        appaserver_user_password,
        post_login_input_form_password ) == 0 )
{
    return appaserver_user_password;
}

enum password_function
    security_password_function(
        appaserver_user_password );

char *security_encrypt_password(
    post_login_input_form_password,
    security_password_function() );

/* Attributes */
POST_LOGIN_INPUT *post_login_input;
char *security_encrypt_password;
boolean password_fail_boolean;
char *reject_index_html_message;
POST_LOGIN_DOCUMENT *post_login_document;
POST_LOGIN_SUCCESS *post_login_success;
```

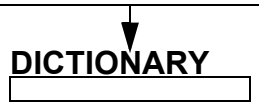


POST_LOGIN (2)

```
/* Driver */
void appaserver_error_argv_file(
    argc,
    argv,
    post_login->
        post_login_input->
        application_name,
    post_login->
        post_login_input->
        login_name );

if ( !post_login->post_login_input->remote_ip_address )
{
    appaserver_error_message_file(
        post_login->
            post_login_input->
            application_name,
        (char *)0 /* login_name */,
        "remote_ip_address not set" );
    exit( 1 );
}

if ( post_login->post_login_document )
{
    printf(
        "%s\n",
        post_login->
            post_login_document->
            html );
}

if ( post_login->password_fail_boolean )
{
    sleep( POST_LOGIN_SLEEP_SECONDS );
}

if ( post_login->post_login_success )
{
    void environment_database_set(
        post_login->
            post_login_input->
            application_name );

    session_insert(
        post_login->
            post_login_success->
            session->
            insert_string,
        post_login->
            post_login_success->
            session->
            insert_system_string );

    if ( post_login->
            post_login_success->
            execute_system_string_frameset )
    {
        system(
            post_login->
            post_login_success->
                execute_system_string_frameset );
    }

    if ( post_login->
            post_login_success->
            email_system_string )
    {
        void appaserver_error_message_file(
            post_login->
                post_login_input->
                application_name,
            post_login->
                post_login_input->
                login_name,
            post_login->
                post_login_success->
                email_link_url );

        void appaserver_error_message_file(
            post_login->
                post_login_input->
                application_name,
            post_login->
                post_login_input->
                login_name,
            post_login->
                post_login_success->
                email_system_string );

        system(
            post_login->
                post_login_success->
                email_system_string );
    }
}
```

POST_LOGIN_INPUT (1)

```
/* Usage */
POST_LOGIN_INPUT *post_login_input_new(
    int argc,
    char **argv );

/* Process */
POST_LOGIN_INPUT *post_login_input_calloc(
    void );

DICTIONARY *post_login_input_dictionary(
    void );

char *post_login_input_application_name(
    argc,
    argv,
    post_login_input_dictionary() );

boolean post_login_input_missing_application_boolean(
    char *post_login_input_application_name() );

if ( post_login_input_missing_application_boolean() )
{
    /* Send argv to admin log file. */
    application_name = APPLICATION_ADMIN_NAME;
    return this;
}

char *post_login_input_remote_ip_address(
    void );

if ( !post_login_input_remote_ip_address() )
{
    return this;
}

boolean post_login_input_invalid_application_boolean(
    char *post_login_input_application_name() );

if ( post_login_input_invalid_application_boolean() )
{
    /* Send argv to admin log file. */
    application_name = APPLICATION_ADMIN_NAME;
    return this;
}

char *post_login_input_login_name(
    APPASERVER_USER_LOGIN_NAME,
    post_login_input_dictionary() );

boolean
    post_login_input_missing_login_name_boolean(
        char *post_login_input_login_name() );

if ( post_login_input_missing_login_name_boolean() )
        return this;

void session_environment_set(
    post_login_input_application_name() );

char *appaserver_parameter_log_directory();

boolean post_login_input_application_exists_boolean(
    post_login_input_application_name(),
    appaserver_parameter_log_directory() );

if ( !post_login_input_application_exists_boolean )
{
    /* Send argv to admin log file. */
    application_name = APPLICATION_ADMIN_NAME;
    return this;
}

boolean post_login_input_email_address_boolean(
    char *post_login_input_login_name() );

if ( !post_login_input_email_address_boolean() )
{
    boolean
        post_login_input_invalid_login_name_boolean(
            char *post_login_input_login_name() );
}

if ( post_login_input_invalid_login_name_boolean() )
        return this;

APPASERVER_USER *appaserver_user =
    appaserver_user_login_fetch(
        post_login_input_login_name(),
        0 /* not fetch_role_name_list */ );

boolean post_login_input_user_not_found_boolean(
    APPASERVER_USER *appaserver_user );

if ( post_login_input_user_not_found_boolean() )
        return this;

boolean post_login_input_public_boolean(
    char *post_login_input_login_name() );

boolean post_login_input_deactivated_boolean(
    char *appaserver_user->password,
    boolean appaserver_user->deactivated_boolean,
    boolean post_login_input_public_boolean() );

if ( post_login_input_deactivated_boolean() )
        return this;

if ( !post_login_input_public_boolean() )
{
    char *post_login_input_form_password(
        DICTIONARY *post_login_input_dictionary() );

    boolean
        post_login_input_form_password_blank_boolean(
            char *post_login_input_form_password() );
}

/* Attributes */
DICTIONARY *dictionary;
char *application_name;
char *remote_ip_address;
boolean missing_application_boolean;
boolean invalid_application_boolean;
boolean application_exists_boolean;
char *login_name;
boolean missing_login_name_boolean;
boolean email_address_boolean;
boolean invalid_login_name_boolean;
boolean user_not_found_boolean;
boolean public_boolean;
boolean deactivated_boolean;
char *form_password;
boolean form_password_blank_boolean;
```

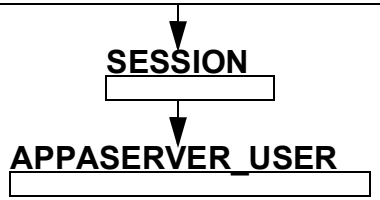


POST_LOGIN_INPUT (2)

```
/* Usage */
char *post_login_input_application_name(
    int argc,
    char **argv,
    DICTIONARY *post_login_dictionary() );

/* Process */
if ( argc == 2 )
{
   char *application_name = argv[ 1 ];
}
else
{
   LIST *key_list = list_new();

   list_set( key_list, “choose_application” );
   list_set( key_list, “application_key” );
   list_set( key_list, “application_name” );

   char *dictionary_list_get(
      post_login_dictionary,
      key_list );

   if ( dictionary_list_get() )
   {
      char *application_name =
         string_low( dictionary_list_get() );
   }
}

/* Usage */
char *post_login_input_login_name(
   const char *APPASERVER_USER_LOGIN_NAME,
   DICTIONARY *post_login_dictionary );

/* Process */
char *dictionary_get(
   APPASERVER_USER_LOGIN_NAME,
   post_login_dictionary );

if ( dictionary_get() )
{
      char *login_name =
         string_low( dictionary_get() );
}

/* Usage */
boolean post_login_input_application_exists_boolean(
   char *post_login_input_application_name,
   char *appaserver_parameter_log_directory );

/* Process */
char *application_log_filename(
   APPLICATION_LOG_EXTENSION,
   post_login_input_application_name,
   appaserver_parameter_log_directory );

boolean application_exists_boolean(
   application_log_filename() );
```

POST_LOGIN_SUCCESS

```
/* Usage */
POST_LOGIN_SUCCESS *post_login_success_new(
    char *post_login_input_application_name,
    char *post_login_Input_login_name,
    boolean
        post_login_input_form_password_blank_boolean,
    boolean post_login_input_email_address_boolean );

/* Process */
POST_LOGIN_SUCCESS *post_login_success_calloc(
    void );

SESSION *session_new(
    post_login_input_application_name,
    post_login_input_login_name,
    1 /* appaserver_user_boolean */ );

char *appaserver_parameter_document_root();

char *post_login_success_output_pipe_string(
    const char *POST_LOGIN_EMAIL_OUTPUT_TEMPLATE,
    char *post_login_input_application_name,
    boolean
        post_login_input_form_password_blank_boolean,
    boolean post_login_input_email_address_boolean,
    char *appaserver_parameter_document_root(),
    char *session_new()->session_key );

char *appaserver_error_filename(
    post_login_input_application_name );

char *execute_system_string_frameset(
    FRAMESET_EXECUTABLE,
    session_new()->session_key,
    post_login_input_login_name,
    post_login_success_output_pipe_string(),
    appaserver_error_filename() );

if (    post_login_input_form_password_blank_boolean
&&  post_login_input_email_address_boolean )
{
    char *appaserver_mailname(
        APPASERVER_MAILNAME_FILESPECIFICATION );

    char *post_login_success_return_address(
        const char *POST_LOGIN_RETURN_USERNAME,
        char *appaserver_mailname() );

    char *post_login_success_email_link_url(
        const char *POST_LOGIN_EMAIL_HTTP_TEMPLATE,
        char *post_login_input_application_name,
        char *session_new()->session_key );

    char *post_login_success_sendmail_message(
        const char *POST_LOGIN_SUBJECT,
        char *post_login_input_login_name,
        char *post_login_success_return_address(),
        char *post_login_success_email_link_url() );

    char *appaserver_parameter_data_directory();

    char *post_login_success_sendmail_filename(
        char *appaserver_parameter_data_directory(),
        pid_t getpid() /* process_id */ );

    char *post_login_success_email_system_string(
        char *post_login_success_sendmail_message(),
        char *post_login_success_sendmail_filename() );

    char *post_login_success_index_html_parameter(
        void );
}

/* Attributes */
SESSION *session;
char *output_pipe_string;
char *appaserver_error_filename;
char *execute_system_string_frameset;
char *return_address;
char *email_link_url;
char *sendmail_message;
char *sendmail_filename;
char *email_system_string;
char *index_html_parameter;
```



POST_LOGIN_DOCUMENT (1)

```
/* Usage */
POST_LOGIN_DOCUMENT *post_login_document_new(
    DICTIONARY *post_login_input_dictionary,
    char *post_login_input_application_name,
    char *index_html_parameter );

/* Process */
POST_LOGIN_DOCUMENT *post_login_document_calloc(
    void );

char *post_login_document_filename(
    post_login_input_dictionary );

enum post_login_document_location_enum
    post_login_document_location_enum_resolve(
        post_login_document_filename() );

char *post_login_document_application_name(
    const char *APPLICATION_TEMPLATE_NAME,
    post_login_input_application_name );

char *post_login_document_filename_location(
    post_login_input_application_name,
    index_html_parameter,
    post_login_document_location_enum_resolve() );

char *application_title_string(
    post_login_document_application_name() );

DOCUMENT *document =
    document_basic_new(
        post_login_document_application_name(),
        (LIST *)0 /* javascript_filename_list */,
        application_title_string() );

char *document_body_open_tag();

char *document_both_close_tag();

char *post_login_document_html(
    post_login_document_location_enum(),
    post_login_document_filename_location(),
    document->html,
    document_body_open_tag(),
    document_both_close_tag() );

/* Attributes */
char *filename;
enum post_login_document_location_enum
    location_enum;
char *application_name;
char *filename_location;
char *application_title_string;
DOCUMENT *document;
char *document_body_open_tag;
char *document_both_close_tag;
char *html;
```

location_html
location_php
location_website

DOCUMENT

POST_LOGIN_DOCUMENT (2)

```
/* Usage */
char *post_login_document_filename(
    DICTIONARY *post_login_input_dictionary );

/* Process */
char *dictionary_get(
    "filename",
    post_login_input_dictionary );

/* Usage */
enum post_login_document_location_enum
    post_login_document_location_enum_resolve(
        char *post_login_document_filename );

/* Process */
if ( !post_login_document_filename )
{
    return location_website;
}

int strcmp(
        post_login_document_filename,
        "index.html" );

if ( strcmp() == 0 )
        return location_html;

int strcmp(
        post_login_document_filename,
        "index.php" );

if ( strcmp() == 0 )
        return location_php;
else
        return location_website;

/* Usage */
char *post_login_document_filename_location(
    char *post_login_input_application_name,
    char *index_html_parameter,
    enum post_login_location_enum
        post_login_document_location_enum );

/* Process */
if (    post_login_document_location_enum ==
        location_html )
{
    string_strcpy(
        static location[],
        index_html_parameter,
        sizeof ( location ) );

    boolean string_exists(
        location,
        "_yn=y" );

    if ( string_exists() )
    {
        *(location + strlen( location ) - 5) = '\0';
    }
}
else
if (    post_login_document_location_enum ==
        location_php )
{
    snprintf(
        static location[],
        sizeof ( location ),
        "/appaserver/%s/index.php?%s",
        post_login_input_application_name,
        index_html_parameter );
}
else
/* Must be location_website */
{
    snprintf(
        static location[],
        sizeof ( location ),
        "/index.php?%s",
        index_html_parameter );
}

/* Usage */
char *post_login_document_html(
    enum post_login_location_enum
        post_login_document_location_enum,
    char *post_login_document_filename_location,
    char *document_html,
    char *document_body_open_tag,
    char *document_close_tag );

/* Process */
if (    post_login_document_location_enum ==
        location_html )
{
    snprintf(
        html[],
        sizeof ( html ),
        "%s\n%s\n"
        "<p>%s.\n"
        "%s",
        document_html,
        document_body_open_tag,
        string_initial_capital(
            destination[],
            post_login_document_filename_location ),
        document_close_tag );
}
else
{
    snprintf(
        html[],
        sizeof ( html ),
        "%s\n%s\n"
        "<script type=\"text/javascript\">\n"
        "window.location = \"%s\"\n"
        "</script>\n"
        "%s",
        document_html,
        document_body_open_tag,
        post_login_document_filename_location,
        document_close_tag );
}
```

APPASERVER_LINK

```
/* Usage */
APPASERVER_LINK *appaserver_link_new(
    char *application_http_prefix(),
    char *application_server_address(),
    char *appaserver_parameter_data_directory(),
    char *filename_stem,
    char *application_name,
    pid_t process_id,
    char *session_key,
    char *begin_date_string,
    char *end_date_string,
    char *extension );

/* Process */
APPASERVER_LINK *appaserver_link_calloc( void );

if ( !filename_stem ) filename_stem = "stem";

char *begin_date_string =
    char *appaserver_link_date_string(
        begin_date_string );

char *end_date_string =
    char *appaserver_link_date_string(
        end_date_string );

APPASERVER_LINK_PROMPT *
    appaserver_link_prompt_new(
        APPASERVER_LINK_PROMPT_DIRECTORY
            /* probably appaserver_data */,
        http_prefix,
        sever_address,
        application_name,
        filename_stem,
        begin_date_string,
        appaserver_link_end_date_string(),
        process_id,
        session_key,
        extension );

char *appaserver_link_filename(
    filename_stem,
    begin_date_string,
    appaserver_link_end_date_string(),
    process_id,
    session_key,
    extension );

char *appaserver_link_tail_half(
    char *application_name,
    char *appaserver_link_filename() );

APPASERVER_LINK_OUTPUT *
    appaserver_link_output_new(
        appaserver_parameter_data_directory,
        appaserver_link_tail_half() );

/* Usage */
char *appaserver_link_filename(
    char *filename_stem,
    char *begin_date_string,
    char *end_date_string,
    pid_t process_id,
    char *session_key,
    char *extension );

/* Process */

/* Usage */
void appaserver_link_pid_filename(
    char **prompt_filename,
    char **output_filename,
    char *application_name,
    char *appaserver_parameter_data_directory(),
    pid_t process_id,
    char *process_name,
    char *extension );

/* Process */
APPASERVER_LINK *appaserver_link_new();

*prompt_filename =
    appaserver_link->
        appaserver_link_output->
        filename;

*output_filename =
    appaserver_link->
        appaserver_link_output->
        filename;

/* Usage */
char *appaserver_link_working_directory(
    char *appaserver_parameter_data_directory(),
    char *application_name );

/* Process */
sprintf(
    working_directory[],
    "%s/%s",
    appaserver_parameter_data_directory,
    application_name );

/* Usage */
char *appaserver_link_anchor_html(
    char *appaserver_link_prompt->filename
        /* prompt_filename */,
    char *target_frame,
    char *prompt_message );

/* Process */
sprintf(
    html[],
    "<br><a href=\"%s\" target=\"%s\">%s</a>\n",
    prompt_filename,
    (target_window) ? target_window : "_new",
    prompt_message );

/* Usage */
char *appaserver_link_date_string(
    char *date_string );

/* Process */
if ( date_string )
{
    char *column( buffer[], 0, date_string );
}

/* Attributes */
char *http_prefix;
char *server_address;
char *appaserver_parameter_data_directory;
char *filename_stem;
char *application_name;
pid_t process_id;
char *session_key;
char *begin_date_string;
char *end_date_string;
char *extension;
APPASERVER_LINK_PROMPT *appaserver_link_prompt;
char *filename;
char *tail_half;
```

APPASERVER_LINK_PROMPT

APPASERVER_LINK_OUTPUT

APPASERVER_LINK_PROMPT

```
/* Usage */
APPASERVER_LINK_PROMPT *
   appaserver_link_prompt_new(
      char *APPASERVER_LINK_PROMPT_DIRECTORY
         /* probably appaserver_data */,
      char *http_prefix,
      char *server_address,
      char *application_name,
      char *filename_stem,
      char *begin_date_string,
      char *end_date_string,
      pid_t process_id,
      char *session_key,
      char *extension );

/* Process */
APPASERVER_LINK_PROMPT *
   appaserver_link_prompt_calloc(
      void );

char *appaserver_link_prompt_link_half(
   char *APPASERVER_LINK_PROMPT_DIRECTORY,
   char *http_prefix,
   char *server_address );

char *appaserver_link_filename(
   filename_stem,
   begin_date_string,
   end_date_string,
   process_id,
   session_key,
   extension );

char *appaserver_link_tail_half(
   application_name,
   appaserver_link_filename() );

char *appaserver_link_prompt_filename(
   char *appaserver_link_prompt_link_half(),
   char *appaserver_link_tail_half() );

/* Attributes */
char *link_half;
char *appaserver_link_filename;
char *appaserver_link_tail_half;
char *filename;
```

APPASERVER_LINK_OUTPUT

```
/* Usage */
APPASERVER_LINK_OUTPUT *
    appaserver_link_output_new(
        char *appaserver_parameter_data_directory(),
        char *appaserver_link_tail_half() );

/* Process */
APPASERVER_LINK_OUTPUT *
    appaserver_link_output_calloc(
        void );

char *appaserver_link_output_filename(
    appaserver_parameter_data_directory,
    appaserver_link_tail_half() );

/* Usage */
char *appaserver_link_output_filename(
    char *appaserver_parameter_data_directory,
    char *appaserver_link_tail_half() );

/* Process */
sprintf(
    char filename[],
    "%s/%s",
    appaserver_parameter_data_directory,
    appaserver_link_tail_half );

/* Attribute */
char *filename;
```

POPULATE_HELPER_PROCESS

```
/* Usage */
POPULATE_HELPER_PROCESS *
    populate_helper_process_new(
        char *
            process_parameter->
                populate_helper_process_name,
        char *document_root_directory,
        char *application_name,
        pid_t getpid() );

/* Input */
char *name;
char *document_root_directory;
char *application_name;
pid_t process_id;

/* Process */
char *populate_helper_process_output_filename(
    char *name,
    char *document_root_directory,
    char *application_name,
    pid_t process_id );

char *populate_helper_process_prompt(
    char *name,
    char *application_name,
    pid_t process_id );
```

QUERY_STATISTIC_ATTRIBUTE

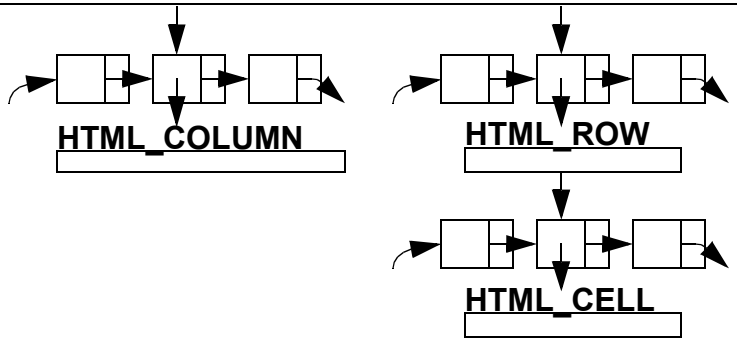
```
/* Usage */
LIST *query_statistic_attribute_list(
    LIST *folder_attribute_number_name_list,
    LIST *folder_attribute_date_name_list,
    LIST *folder_attribute_date_time_name_list,
    LIST *query_statistic_record_list );

QUERY_STATISTIC_ATTRIBUTE *
    query_statistic_attribute_new(
        char *folder_attribute_name,
        boolean is_date,
        int piece_offset,
        LIST *query_statistic_record_list );

/* Process */
QUERY_STATISTIC_AGGREGATE *
    query_statistic_aggregate =
        query_statistic_aggregate_new(
            is_date,
            piece_offset,
            query_statistic_record_list );

char *query_statistic_attribute_html(
    char *folder_attribute_name,
    boolean is_date,
    QUERY_STATISTIC_AGGREGATE *
        query_statistic_aggregate );
```

QUERY_STATISTIC_AGGREGATE

```
/* Usage */
QUERY_STATISTIC_AGGREGATE *
    query_statistic_aggregate_new(
        boolean is_date,
        int piece_offset,
        LIST *record_list );

/* Attributes */
unsigned long count;
double sum;
double average;
double minimum;
double maximum;
double range;
double standard_deviation;
double standard_error;
double percent_missing;
char *begin_date;
char *end_date;

/* Process */
char *aggregate_filename =
    timlib_temp_filename( (char *)0 /* key */ );

FILE *output_pipe =
    query_statistic_aggregate_output_pipe(
        boolean is_date,
        char *aggregate_filename );

void query_statistic_aggregate_generate(
        FILE *output_pipe,
        int piece_offset,
        LIST *record_list );

pclose( output_pipe );

QUERY_STATISTIC_AGGREGATE *
    query_statistic_aggregate_read(
        boolean is_date,
        char *aggregate_filename );

timlib_remove_file( aggregate_filename );
```

HTML_TABLE

```
/* Usage */
HTML_TABLE *html_table_new(
    char *title,
    char *sub_title,
    char *sub_sub_title );

/* Process */
HTML_TABLE *html_table_calloc(
    void );

/* Driver */
void html_table_output(
    HTML_TABLE *html_table,
    int HTML_TABLE_ROWS_BETWEEN_HEADING );

/* Process */
void html_table_title_output(
    char *title,
    char *sub_title,
    char *sub_sub_title );

void html_column_list_output(
    html_table->column_list );

for HTML_ROW *html_row in html_table->row_list
{
    if (++row_count ==
        HTML_TABLE_ROWS_BETWEEN_HEADING )
    {
        html_column_list_output(
            html_table->column_list );

        row_count = 0;
    }

    void html_row_output(
        html_table->column_list,
        html_row );
}

char *html_table_close_tag( void );

printf( "%s\n", html_table_close_tag() );

/* Driver */
void html_table_list_output(
    LIST *html_table_list,
    int HTML_TABLE_ROWS_BETWEEN_HEADING );

/* Process */
for HTML_TABLE *html_table in html_table_list
{
    html_table_output(
        html_table,
        HTML_TABLE_ROWS_BETWEEN_HEADING );
}

/* Driver */
void html_string_list_stdout(
    int HTML_TABLE_QUEUE_TOP_BOTTOM,
    char *title,
    char *heading_string,
    LIST *delimited_string_list );

/* Process */
char *html_table_system_string(
    HTML_TABLE_QUEUE_TOP_BOTTOM,
    '^' /* delimiter */,
    title,
    heading_string );

FILE *appaserver_output_pipe(
    html_table_system_string() ));

for char *delimited_string in delimited_string_list
{
    fprintf(    appaserver_output_pipe(),
                "%s\n",
                delimited_string );
}

pclose( appaserver_output_pipe() );

/* Usage */
char *html_table_system_string(
    int HTML_TABLE_QUEUE_TOP_BOTTOM,
    char delimiter,
    char *title,
    char *heading_string );

/* Process */
snprintf(system_string[],
            sizeof ( system_string ),
            "queue_top_bottom_lines.e %d |"
            "html_table.e '^^%s' '%s' '%c'",
            HTML_TABLE_QUEUE_TOP_BOTTOM,
            (title) ? title : "",
            heading_string,
            delimiter);

/* Attributes */
char *title;
char *sub_title;
char *sub_sub_title;

/* Set externally */
LIST *column_list;
LIST *row_list;
```



HTML_COLUMN

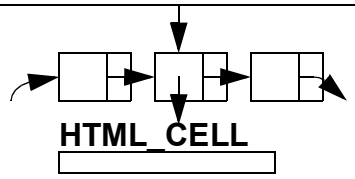
```
/* Usage */
HTML_COLUMN *html_column_new(
    char *heading,
    boolean right_justify_boolean );

/* Process */
HTML_COLUMN *html_column_calloc(
    void );

/* Usage */
LIST *html_column_right_justify_list(
    LIST *heading_list );

/* Process */
LIST *column_list = list_new();

for char *heading in heading_list
{
    list_set(
        column_list,
        html_column_new(
            heading,
            1 /* right_justify_boolean */ ) );
}

/* Usage */
LIST *html_column_list(
    LIST *heading_list,
    LIST *justify_list );

/* Process */
LIST *list = list_new();

if ( list_length( justify_list ) ) list_rewind( justify_list );

for char *heading in heading_list
{
    if ( list_length( justify_list ) )
    {
        char *justify = list_get( justify_list );

        boolean html_column_right_justify_boolean(
            char *justify );
    }

    HTML_COLUMN *html_column_new(
        heading,
        html_column_right_justify_boolean() );

    list_set( list, html_column_new() );

    if ( list_length( justify_list ) ) list_next( justify_list );
}

/* Driver */
void html_column_list_output(
    LIST *column_list );

/* Process */
printf( "<tr>\n" );

for HTML_COLUMN *html_column in column_list
{
    char *html_column_tag(
        html_column->heading );

    printf( "%s\n", html_column_tag() );
}

/* Usage */
char *html_column_tag(
    char *heading );

/* Process */
if ( !heading ) heading = "";

if (    string_character_exists( heading, '&' )
||      string_character_exists( heading, ';' ) )
{
    strcpy( buffer[], heading );
}
else
{
    string_initial_capital( buffer[], heading );
}

sprintf( static tag[], "<td>%s", buffer );

/* Attributes */
char *heading;
boolean right_justify_boolean;
```

HTML_ROW

```
/* Usage */
HTML_ROW *html_row_new(
    LIST *cell_list );

/* Process */
HTML_ROW *html_row_calloc(
    void );

/* Driver */
void html_row_output(
    LIST *column_list,
    HTML_ROW *html_row );
```

```
/* Process */
list_rewind( column_list );

printf( "<tr>\n" );

for HTML_CELL *html_cell in html_row->cell_list
{
    html_cell_output(
        list_get( column_list ),
        html_cell,
        html_row->background_shaded_boolean );
    list_next( column_list );
}
```

```
/* Attribute */
LIST *cell_list;

/* Populated externally */
boolean background_shaded_boolean;
```

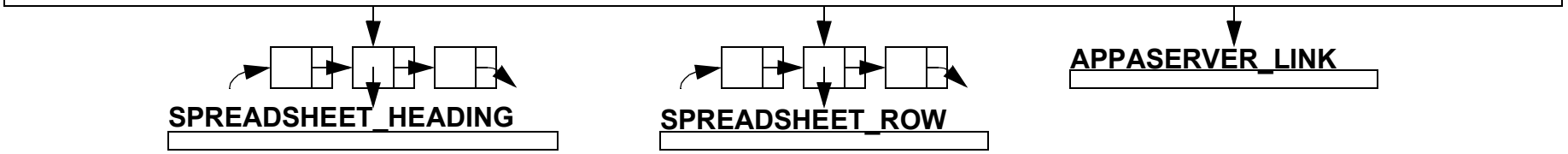


HTML_CELL

```
/* Usage */
HTML_CELL *html_cell_new(
    char *data,
    boolean large_boolean,
    boolean bold_boolean );

/* Process */
HTML_CELL *html_cell_calloc(
    void );

/* Usage */
void html_cell_datum_set(
    LIST *cell_list,
    char *datum,
    boolean large_boolean,
    boolean bold_boolean );

/* Process */
HTML_CELL *html_cell =
    html_cell_new(
        datum,
        large_boolean,
        bold_boolean );

list_set(
    cell_list,
    html_cell );

/* Usage */
void html_cell_data_list_set(
    LIST *cell_list,
    LIST *data_list );

/* Process */
for char *datum in data_list
{
    html_cell_datum_set(
        cell_list,
        datum,
        0 /* not large_boolean */,
        0 /* not bold_boolean */ );

/* Usage */
LIST *html_cell_list(
    LIST *data_list );

/* Process */
LIST *cell_list = list_new();

for char *datum in data_list
{
    list_set(
        cell_list,
        html_cell_new(
            datum,
            0 /* not large_boolean */,
            0 /* not bold_boolean */ ) );
}

/* Driver */
void html_cell_output(
    HTML_COLUMN *html_column,
    HTML_CELL *html_cell,
    boolean background_shaded_boolean );

/* Process */
char *html_cell_datum_tag(
    html_cell->datum,
    html_cell->large_boolean,
    html_cell->bold_boolean,
    html_column->right_justify_boolean,
    background_shaded_boolean );

printf "%s\n", html_cell_data_tag() );

/* Usage */
char *html_cell_datum_tag(
    char *datum,
    boolean large_boolean,
    boolean bold_boolean,
    boolean right_justify_boolean,
    boolean background_shaded_boolean );

/* Process */
char tag[];
char *ptr = tag;

if ( !datum ) datum = "";

ptr += sprintf( ptr, "<td" );

if ( background_shaded_boolean )
{
    ptr += sprintf( ptr, " bgcolor=\"#CCCCCC\"" );
}

if ( *datum && right_justify_boolean )
{
    ptr += sprintf( ptr, " align=right" );
}

ptr += sprintf( ">" );

if ( *datum && large_boolean )
{
    ptr += sprintf( ptr, "<big>" );
}

if ( *datum && bold_boolean )
{
    ptr += sprintf( ptr, "<b>" );
}

ptr += sprintf( ptr, "%s", datum );

if ( *datum && bold_boolean )
{
    ptr += sprintf( ptr, "</b>" );
}

if ( *datum && large_boolean )
{
    ptr += sprintf( ptr, "</big>" );
}

/* Attributes */
char *datum;
boolean large_boolean;
boolean bold_boolean;
```

SPREADSHEET

```
/* Usage */
SPREADSHEET *spreadsheet_new(
    char *application_name,
    char *data_directory,
    pid_t process_id );

/* Process */
SPREADSHEET *spreadsheet_calloc(
    void );

LIST *column_list = list_new();
LIST *row_list = list_new();

APPASERVER_LINK *appaserver_link =
   appaserver_link_new(
      application_http_prefix(
         application_ssl_support_boolean(
            application_name ) ),
      application_server_address(),
      data_directory,
      "spreadsheet" /* filename_stem */,
      application_name,
      process_id,
      (char *)0 /* session_key */,
      (char *)0 /* begin_date_string */,
      (char *)0 /* end_date_string */,
      "csv" /* extension */ );

char *output_filename =
   appaserver_link->
      appaserver_link_output->
      filename;

char *appaserver_link_anchor_html(
   appaserver_link->
      appaserver_link_prompt->
      filename,
   "_new" /* target_window */,
   "Link to spreadsheet" /* prompt_message */ );

/* Driver */
void spreadsheet_output(
    SPREADSHEET *spreadsheet,
    FILE *output_file );

/* Process */
void spreadsheet_column_list_output(
    spreadsheet->column_list,
    output_file );

for SPREADSHEET_ROW *
    spreadsheet_row in
        spreadsheet->row_list
{
    void spreadsheet_row_output(
        spreadsheet_row,
        output_file );
}

/* Attributes */
LIST *column_list;
LIST *row_list;
APPASERVER_LINK *appaserver_link;
char *output_filename;
char *appaserver_link_anchor_html;
```

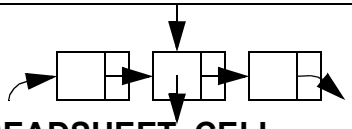


SPREADSHEET_HEADING

```
/* Usage */
SPREADSHEET_HEADING *
    spreadsheet_heading_new(
        char *heading_string );

/* Process */
SPREADSHEET_HEADING *
    spreadsheet_heading_calloc(
        void );

/* Usage */
void spreadsheet_column_list_output(
        LIST *column_list,
        FILE *output_file );

/* Process */
for SPREADSHEET_HEADING *
    spreadsheet_heading in
        column_list
{
    if ( !list_at_first( column_list ) )
        fprintf( output_file, "," );

    fprintf(
        output_file,
        "\"%s\"",
        string_initial_capital(
            buffer[],
            spreadsheet_heading->heading_string ) );
}

fprintf( output_file, "\n" );

/* Attribute */
char *heading_string;
```

SPREADSHEET_ROW

```
/* Usage */
SPREADSHEET_ROW *spreadsheet_row_new( void );

/* Process */
SPREADSHEET_ROW *spreadsheet_row_calloc(
    void );

LIST *cell_list = list_new();

/* Usage */
void spreadsheet_row_output(
    SPREADSHEET_ROW *spreadsheet_row,
    FILE *output_file );

/* Process */
for SPREADSHEET_CELL *
    spreadsheet_cell in
        spreadsheet_row->cell_list
{
    void spreadsheet_cell_output(
            spreadsheet_cell,
            output_file );

    if ( !list_at_first( spreadsheet_row->cell_list ) )
        fprintf( output_file, "," );
}

fprintf( output_file, "\n" );

/* Attribute */
LIST *cell_list;
```



SPREADSHEET_CELL

```
/* Usage */
SPREADSHEET_CELL *spreadsheet_cell_new(
    char *data,
    boolean string_boolean );

/* Process */
SPREADSHEET_CELL *spreadsheet_cell_calloc(
    void );

/* Usage */
void spreadsheet_cell_output(
    SPREADSHEET_CELL *spreadsheet_cell,
    FILE *output_file );

/* Process */
if ( spreadsheet_cell->string_boolean )
    fprintf( output_file, "%c", '"' );

fprintf( output_file, "%s", spreadsheet_cell->data );

if ( spreadsheet_cell->string_boolean )
    fprintf( output_file, "%c", '"' );

/* Attributes */
char *data;
boolean string_boolean;
```

LATEX (1)

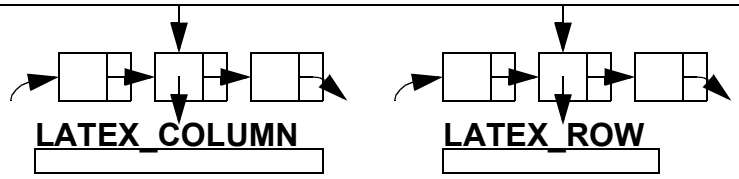
```
/* Usage */
LATEX *latex_new(
    char *tex_filename,
    char *working_directory,
    boolean landscape_boolean,
    char *logo_filename );

/* Process */
LATEX *latex_calloc( void );

char *latex_directory_filename(
    char *tex_filename,
    char *working_directory );

char *latex_documentclass(
    landscape_boolean );

char *latex_usepackage();

char *date_now16( date_utc_offset() );

char *latex_footer_declaration(
    date_now16(),
    LATEX_PBJ );

char *latex_begin_document();

boolean file_exists_boolean( logo_filename );

if ( file_exists_boolean() )
{
    char *latex_logo_display( logo_filename );
}

char *latex_end_document();

/* Usage */
void latex_tex2pdf(
    char *tex_filename,
    char *working_directory );

/* Process */
chdir( working_directory );

snprintf(
    system_string[],
    sizeof ( system_string ),
    "pdflatex %s 2>&1",
    tex_filename );

/* Need to run it four times because */
/* header widths may need adjustments. */
system( system_string );
system( system_string );
system( system_string );
system( system_string );

/* Usage */
FILE *latex_output_file(
    char *latex->directory_filename );

/* Process */
FILE *appaserver_output_file(
    directory_filename );

/* Usage */
void latex_output_document_head(
    char *latex_documentclass,
    char *latex_usepackage,
    char *latex_footer_declaration,
    char *latex_begin_document,
    char *latex_logo_display,
    FILE *latex_output_file() );

/* Process */
char *latex_document_head_display(
    latex_documentclass,
    latex_usepackage,
    latex_footer_declaration,
    latex_begin_document,
    latex_logo_display );

fprintf(
    latex_output_file,
    "%s\n",
    latex_document_head_display() );

/* Usage */
char *latex_document_head_display(
    char *latex_documentclass,
    char *latex_usepackage,
    char *latex_footer_declaration,
    char *latex_begin_document,
    char *latex_logo_display );

/* Process */
char display[];
char *ptr = display;

ptr += sprintf( ptr,
                "%s\n",
                latex_documentclass );

ptr += sprintf( ptr,
                "%s\n",
                latex_usepackage );

if ( latex_footer_declaration )
{
    ptr += sprintf( ptr,
                    "%s\n",
                    latex_footer_declaration );
}

ptr += sprintf( ptr,
                "%s\n",
                latex_begin_document );

if ( latex_logo_display )
{
    ptr += sprintf( ptr,
                    "%s\n",
                    latex_logo_display );
}

/* Usage */
void latex_output_document_tail(
    char *latex->end_document,
    FILE *latex_output_file() );

/* Process */
fprintf(    latex_output_file,
            "%s\n",
            end_document );

/* Attributes */
char *directory_filename;
char *documentclass;
char *usepackage;
char *footer_declaration;
char *begin_document;
char *logo_display;
char *end_document;
```

LATEX (2)

```
/* Usage */
char *latex_documentclass(
    boolean landscape_boolean );

/* Process */
if ( landscape_boolean )
{
    return
    "\\documentclass[a4paper,landscape]{report}\n"
    "\\special{landscape}";
}
else
{
    return
    "\\documentclass[a4paper]{report}";
}

/* Usage */
char *latex_usepackage( void );

/* Process */
return
"\\usepackage{graphics}\n"
"\\usepackage{fancyhdr}\n"
"\\usepackage[labelformat=empty]{caption}\n"
"\\usepackage{longtable}\n"
"\\usepackage[\n"
"   margin=1.5cm,\n"
"   vmargin=1.5cm,\n"
"   nohead]{geometry}";

/* Usage */
char *latex_footer_declaration(
    char *date_now16(),
    char *LATEX_PBJ );

/* Process */
sprintf(    static footer_declaration[],
            "\\pagestyle{fancy}\n"
            "\\fancyhf{}\n"
            "\\cfoot{%s (%s)}\n"
            "\\rfoot{Page \\thepage}",
            date_now16,
            LATEX_PBJ );

/* Usage */
char *latex_begin_document(
    void );

/* Process */
return "\\begin{document}"

/* Usage */
char *latex_logo_display(
    char *logo_filename );

/* Process */
sprintf(    static logo_display[],
            "\\begin{center}\n"
            "\\includegraphics{%s}\n"
            "\\end{center}",
            logo_filename );

/* Usage */
char *latex_end_document(
    void );

/* Process */
return "\\end{document}"
```

LATEX_TABLE (1)

```
/* Usage */
LATEX_TABLE *latex_table_new(
    char *report_title,
    char *table_title,
    LIST *latex_column_list,
    LIST *latex_row_list );

/* Process */
LATEX_TABLE *latex_table_calloc(
    void );

char *latex_table_begin_longtable();

char *latex_column_header_format_line(
    latex_column_list );

char *latex_table_caption_display(
    report_title,
    table_title );

char *latex_column_header_text_line(
    latex_column_list );

char *latex_table_footer_display();

char *latex_table_end_longtable();

/* Attibutes */
LIST *latex_column_list;
LIST *latex_row_list;
char *begin_longtable;
char *latex_column_header_format_line;
char *caption_display;
char *latex_column_header_text_line;
char *footer_display;
char *end_longtable;
```



```
/* Usage */
char *latex_table_begin_longtable(
    void );

/* Process */
return "\\begin{longtable}";

/* Usage */
char *latex_table_caption_display(
    char *report_title,
    char *table_title );

/* Process */
char *latex_table_caption_string(
    char *report_title,
    char *table_title );

char *string_initial_capital(
    buffer[],
    latex_table_caption_string() );

snprintf(
    caption_display[],
    sizeof ( caption_display ),
    "\\caption {%s}\\\\\n"
    "\\hline",
    string_initial_capital() );

/* Usage */
char *latex_table_footer_display( void );

/* Process */
return
"\\hline \\hline \\endhead\n"
"\\hline \\multicolumn{2}{r}"
"{\\textit{Continued on next page}} \\\\\n"
"\\hline\n"
"\\endfoot\n"
"\\endlastfoot";

/* Usage */
char *latex_table_end_longtable(
    void );

/* Process */
return
"\n"
"\\hline\n"
"\\end{longtable}";

/* Usage */
void latex_table_output_head(
    char *latex_table_begin_longtable,
    char *latex_column_header_format_line,
    char *latex_table_caption_display,
    char *latex_column_header_text_line,
    char *latex_table_footer_display,
    FILE *latex_output_file() );

/* Process */
char *latex_table_head_display(
    latex_table_begin_longtable,
    latex_column_header_format_line,
    latex_table_caption_display,
    latex_column_header_text_line,
    latex_table_footer_display );

fprintf(
    latex_output_file,
    "%s\n",
    latex_table_head_display() );

/* Usage */
char *latex_table_head_display(
    char *latex_table_begin_longtable,
    char *latex_column_header_format_line,
    char *latex_table_caption_display,
    char *latex_column_header_text_line,
    char *latex_table_footer_display );

/* Process */
char display[];

char *ptr = display;

ptr += sprintf( ptr,
                "%s",
                latex_table_begin_longtable );

ptr += sprintf( ptr,
                "%s\n",
                latex_column_header_format_line );

ptr += sprintf( ptr,
                "%s\n",
                latex_table_caption_display );

ptr += sprintf( ptr,
                "%s\n",
                latex_column_header_text_line );

ptr += sprintf( ptr,
                "%s\n",
                latex_table_footer_display );

/* Usage */
void latex_table_output_row_list(
    LIST *latex_table->latex_row_list /* freed */,
    FILE *latex_output_file() );

/* Process */
void latex_row_list_output(
    latex_row_list /* freed */,
    latex_output_file );

/* Usage */
void latex_table_output_tail(
    char *latex_table->end_longtable,
    FILE *latex_output_file() );

/* Process */
fprintf(    latex_output_file,
            "%s\n",
            end_longtable )
```

LATEX_TABLE (3)

```
/* Driver */
void latex_table_output(
    char *tex_filename,
    char *tex_anchor_html,
    char *pdf_anchor_html,
    char *working_directory,
    LATEX *latex,
    LATEX_TABLE *latex_table );

/* Process */
FILE *latex_output_file(
    latex->directory_filename );

void latex_output_document_head(
    latex->documentclass,
    latex->usepackage,
    latex->footer_declaration,
    latex->begin_document,
    latex->logo_display,
    latex_output_file() );

void latex_table_output_head(
    latex_table->begin_longtable,
    latex_table->latex_column_header_format_line,
    latex_table->caption_display,
    latex_table->latex_column_header_text_line,
    latex_table->footer_display,
    latex_output_file() );

void latex_table_output_row_list(
    latex_table->row_list /* freed */,
    latex_output_file() );

void latex_table_output_tail(
    latex_table->end_longtable,
    latex_output_file() );

void latex_output_document_tail(
    latex->end_document,
    latex_output_file() );

void fclose( latex_output_file() );

void latex_tex2pdf(
    tex_filename,
    working_directory );

printf(
    "%s<br>%s\n",
    pdf_anchor_html,
    tex_anchor_html );

/* Driver */
void latex_table_list_output(
    char *tex_filename,
    char *tex_anchor_html,
    char *pdf_anchor_html,
    char *working_directory,
    LATEX *latex,
    LIST *latex_table_list );

/* Process */
FILE *latex_output_file(
    latex->directory_filename );

void latex_output_document_head(
    latex->documentclass,
    latex->usepackage,
    latex->footer_declaration,
    latex->begin_document,
    latex->logo_display,
    latex_output_file() );

for LATEX_TABLE *latex_table in latex_table_list
{
    void latex_table_output_head(
        latex_table->begin_longtable,
        latex_table->latex_column_header_format_line,
        latex_table->caption_display,
        latex_table->latex_column_header_text_line,
        latex_table->footer_display,
        latex_output_file() );

    void latex_table_output_row_list(
        latex_table->row_list /* freed */,
        latex_output_file() );

    void latex_table_output_tail(
        latex_table->end_longtable,
        latex_output_file() );
}

void latex_output_document_tail(
    latex->end_document,
    latex_output_file() );

void fclose( latex_output_file() );

void latex_tex2pdf(
    tex_filename,
    working_directory );

printf(
    "%s<br>%s\n",
    pdf_anchor_html,
    tex_anchor_html );
```

LATEX_COLUMN (1)

```
/* Usage */
LATEX_COLUMN *latex_column_new(
    char *heading_string,
    enum latex_column_enum latex_column_enum,
    int float_decimal_count,
    char *paragraph_size,
    boolean first_column_boolean );

/* Process */
LATEX_COLUMN *latex_column_calloc(
    void );

boolean latex_column_right_justify_boolean(
    latex_column_enum );

char *latex_column_display(
    char *heading_string );

/* Usage */
boolean latex_column_right_justify_boolean(
    enum latex_column_enum latex_column_enum );

/* Process */
if (latex_column_enum == latex_column_integer
||  latex_column_enum == latex_column_float )
{
    return 1;
}
else
{
    return 0;
}

/* Usage */
char *latex_column_display(
    char *heading_string );

/* Process */
char *string_initial_capital(
    display[],
    heading_string );

char *strdup( display[] );

/* Usage */
LIST *latex_column_float_list(
    LIST *heading_string_list,
    boolean first_column_boolean,
    boolean dollar_sign_boolean );

/* Process */
LIST *list = list_new();

for char *heading_string in heading_string_list
{
    LATEX_COLUMN *latex_column =
        latex_column_new(
            heading_string,
            latex_column_float /* latex_column_enum */,
            2 /* float_decimal_count */,
            (char *)0 /* paragraph_size */,
            first_column_boolean );

    latex_column->dollar_sign_boolean =
        dollar_sign_boolean;

    list_set(
        list,
        latex_column );

    first_column_boolean = 0;
}

/* Usage */
LIST *latex_column_text_list(
    LIST *heading_string_list,
    boolean first_column_boolean,
    boolean right_justify_boolean );

/* Process */
LIST *list = list_new();

for char *heading_string in heading_string_list
{
    LATEX_COLUMN  *latex_column  =
        latex_column_new(
            heading_string,
            latex_column_text /* latex_column_enum */,
            0 /* float_decimal_count */,
            (char *)0 /* paragraph_size */,
            first_column_boolean );

    latex_column->right_justify_boolean =
        right_justify_boolean;

    list_set(
        list,
        latex_column );

    first_column_boolean = 0;
}

/* Attributes */
char *heading_string;
enum latex_column_enum latex_column_enum;
int float_decimal_count;
char *paragraph_size;
boolean first_column_boolean;
boolean right_justify_boolean;
char *display;

/* Set externally */
boolean dollar_sign_boolean;
```

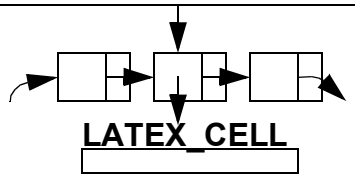
latex_column_text
latex_column_integer
latex_column_float

LATEX_COLUMN (2)

```
/* Usage */
char *latex_column_header_format_line(
    LIST *latex_column_list );

/* Process */
char format_line[];
char *ptr = format_line;

ptr += sprintf( ptr, "{|" );

for LATEX_COLUMN *latex_column in latex_column_list
{
    if ( latex_column->paragraph_size )
    {
        /*
        ptr += sprintf( ptr,
        ">{\\RaggedRight\\arraybackslash}p{%s}|",
        latex_column->paragraph_size );
        */

        ptr += sprintf( ptr,
        "p{%s}|",
        latex_column->paragraph_size );
    }
    else
    if ( latex_column->right_justify_boolean )
    {
        ptr += sprintf( ptr, "r|" );
    }
    else
    {
        ptr += sprintf( ptr, "l|" );
    }
}

ptr += sprintf( ptr, "}" );

/* Usage */
char *latex_column_header_text_line(
    LIST *latex_column_list );

/* Process */
char text_line[];
char *ptr = text_line;

for LATEX_COLUMN *latex_column in latex_column_list
{
    boolean list_first_boolean( latex_column_list );

    if ( !list_first_boolean() )
    {
        ptr += sprintf( ptr, " & " );
    }

    ptr += sprintf( ptr,
        "%s",
        latex_column->display );
}

ptr += sprintf( ptr, "\\\\[0.5ex]" );

/* Usage */
LATEX_COLUMN *latex_column_seek(
    LIST *latex_column_list,
    char *heading_string );

/* Process */
```

LATEX_ROW

```
/* Usage */
LATEX_ROW *latex_row_new(
    LIST *cell_list );

/* Process */
LATEX_ROW *latex_row_calloc(
    void );

/* Usage */
char latex_row_list_display(
    LIST *latex_row_list /* freed */ );

/* Process */
char display[];
char *ptr = display;

for LATEX_ROW *latex_row in latex_row_list
{
    if ( latex_row->preceed_double_line_boolean )
    {
        ptr +=
            sprintf(
                ptr,
                "\\hline \\hline\n" );
    }

    char *latex_cell_list_display(
        latex_row->cell_list );

    ptr +=
        sprintf(
            ptr,
            "%s\\\\\n",
            latex_cell_list_display() );

    free( latex_cell_list_display() );

    latex_cell_list_free(
        latex_row->cell_list );
}

void list_free( latex_row_list );

/* Driver */
void latex_row_list_output(
    LIST *latex_row_list /* freed */,
    FILE *latex_output_file() );

/* Process */
char *latex_row_list_display(
    latex_row_list /* freed */ );

fprintf(
    latex_output_file,
    "%s\n",
    latex_row_list_display() );

free( latex_row_list_display() );

/* Attributes */
LIST *cell_list;

/* Set externally */
boolean preceed_double_line_boolean;
```



LATEX_CELL (1)

```
/* Usage */
LATEX_CELL *latex_cell_float_new(
    LATEX_COLUMN *latex_column,
    double value );

/* Process */
char *float_string(
    value,
    latex_column->float_decimal_count );

LATEX_CELL *latex_cell_small_new(
    latex_column,
    strdup( float_string() /* datum */ );

/* Usage */
LATEX_CELL *latex_cell_small_new(
    LATEX_COLUMN *latex_column,
    char *datum );

/* Process */
LATEX_CELL *latex_cell_new(
    latex_column,
    0 /* not first_row_boolean */,
    datum,
    0 /* not large_boolean */,
    0 /* not bold_boolean */ );

/* Usage */
LATEX_CELL *latex_cell_new(
    LATEX_COLUMN *latex_column,
    boolean first_row_boolean,
    char *datum,
    boolean large_boolean,
    boolean bold_boolean );

/* Process */
LATEX_CELL *latex_cell_calloc( void );

char *latex_cell_column_delimiter(
    latex_column->first_column_boolean );

char *latex_cell_markup(
    large_boolean,
    bold_boolean )

boolean latex_cell_dollar_sign_boolean(
    boolean latex_column->dollar_sign_boolean,
    boolean first_row_boolean );

char *latex_cell_formatted_datum(
    datum,
    latex_column->latex_column_enum,
    latex_column->float_decimal_count,
    latex_cell_dollar_sign_boolean() );

char *latex_cell_display(
    latex_cell_column_delimiter(),
    latex_cell_markup(),
    latex_cell_formatted_datum() );

/* Attribute */
char *display;
```

LATEX_CELL (2)

```
/* Usage */
char *latex_cell_column_delimiter(
    boolean first_column_boolean );

/* Process */
if ( first_column_boolean )
    return "";
else
    return " &";

/* Usage */
char *latex_cell_markup(
    boolean large_boolean,
    boolean bold_boolean );

/* Process */
static char markup[];
char *ptr = markup;

*ptr = '\0';

if ( large_boolean ) ptr += sprintf( ptr, "\\large " );

if ( bold_boolean ) ptr += sprintf( ptr, "\\bf " );

/* Usage */
char *latex_cell_formatted_datum(
    char *datum,
    enum latex_column_enum latex_column_enum,
    int float_decimal_count,
    boolean dollar_sign_boolean );

/* Process */
if ( !datum || !*datum ) return "";

if ( latex_column_enum == latex_column_text )
{
    datum = char *latex_cell_escape( datum );
}

if ( latex_column_enum == latex_column_float )
{
    char *latex_cell_dollar_sign(
        boolean dollar_sign_boolean );

    double string_atof( datum );

    char *string_commas_double(
        string_atof(),
        float_decimal_count );

    sprintf( static formatted_datum[],
        "%s%s",
        latex_cell_dollar_sign(),
        string_commas_double() );
}
else
{
    return datum;
}

/* Usage */
char *latex_cell_display(
    char *latex_cell_column_delimiter(),
    char *latex_cell_markup(),
    char *latex_cell_formatted_datum() );

/* Process */
snprintf(
    display[],
    sizeof ( display ),
    "%s %s %s",
    latex_cell_column_delimiter,
    latex_cell_markup,
    latex_cell_formatted_datum );

/* Usage */
char *latex_cell_escape(
    char *datum );

/* Process */
string_strcpy( static escape[], datum );

string_search_replace( escape, "$", "\\$" );
string_search_replace( escape, "&", "\\&" );
string_search_replace( escape, "%", "\\%" );
string_search_replace( escape, "#", "\\#" );
string_search_replace( escape, "_", "\\_" );
```

LATEX_CELL (3)

```
/* Usage */
LIST *latex_cell_list(
    const char *calling_function,
    LIST *latex_column_list,
    LIST *data_list );

/* Process */
LIST *list = list_new();

/* No list_rewind( latex_column_list ) */

for char *datum in data_list
{
    LATEX_COLUMN *latex_column =
        list_get( latex_column_list );

    void list_next( latex_column_list );

    LATEX_CELL *latex_cell_small_new(
        latex_column,
        datum );

    list_set( list, latex_cell_small_new() );
}

/* Usage */
char *latex_cell_list_display(
    LIST *latex_cell_list );

/* Process */
char display[];
char *ptr = display;

for LATEX_CELL *latex_cell in latex_cell_list
{
    ptr += sprintf( ptr,
        "%s\n",
        latex_cell->display );
}

/* Usage */
void latex_cell_list_free(
    LIST *latex_cell_list );

/* Process */
for LATEX_CELL *latex_cell in latex_cell_list
{
    free( latex_cell->display );
}
list_free( latex_cell_list );
```

MERGE_PURGE

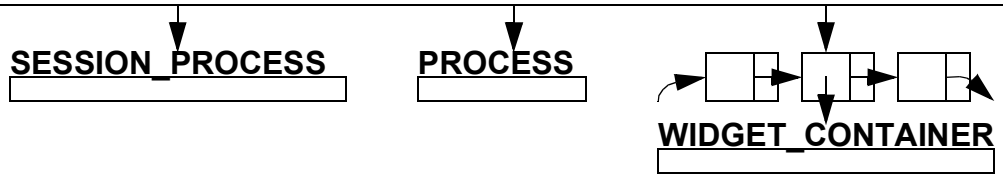
```
/* Usage */
char *merge_purge_hypertext_reference(
    char *executable,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *process_name,
    char *folder_name );
```

MERGE_PURGE_CHOOSE (1)

```
/* Usage */
MERGE_PURGE_CHOOSE *merge_purge_choose_new(
    int argc,
    char **argv,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *process_name );

/* Process */
MERGE_PURGE_CHOOSE *merge_purge_choose_calloc(
    void );

/* Sets appaserver environment and outputs argv. */
/* Each parameter is security inspected. */
/* Any error will exit( 1 ). */
SESSION_PROCESS *session_process =
    session_process_integrity_exit(
        argc,
        argv,
        application_name,
        session_key,
        login_name,
        role_name,
        process_name );

char *merge_purge_choose_title_string(
    char *process_name );

char *merge_purge_choose_sub_sub_title_string(
    void );

PROCESS *process_fetch(
    process_name,
    (char *)0 /* document_root */
    (char *)0 /* application_relative_source_directory */,
    0 /* not check_executable_inside_filesystem */,
    (char *)0 /* appaserver_parameter_mount_point */ );

char *document_body_notepad_html(
    process_fetch->notepad );

char *document_body_html(
    (char *)0 /* menu_html */,
    merge_purge_choose_title_string(),
    (char *)0 /* sub_title_string */,
    merge_purge_choose_sub_sub_title_string(),
    (char *)0 /* body_onload_open_tag */,
    (char *)0 /* document_body_hide_preload_html */,
    document_body_notepad_html() );

char *merge_purge_choose_role_folder_where(
    const char *ROLE_FOLDER_TABLE,
    const char *RELATION_TABLE,
    const char *APPASERVER_UPDATE_STATE,
    char *role_name );

LIST *merge_purge_choose_role_folder_name_list(
    char *ROLE_FOLDER_TABLE,
    char *merge_purge_choose_role_folder_where() );

LIST *merge_purge_choose_widget_container_list(
    application_name,
    session_key,
    login_name,
    role_name,
    process_name,
    merge_purge_choose_role_folder_name_list() );

char *widget_container_list_html(
    (char *)0 /* state */,
    -1 /* row_number to omit index suffix */,
    (char *)0 /* background_color */,
    widget_container_list );

char *document_form_html(
    (char *)0 /* document_html */,
    (char *)0 /* document_head_html */,
    (char *)0 /* document_head_close_tag */,
    document_body_html(),
    widget_container_list_html(),
    document_body_close_tag(),
    document_close_tag() );

free( widget_container_list_html() );
```

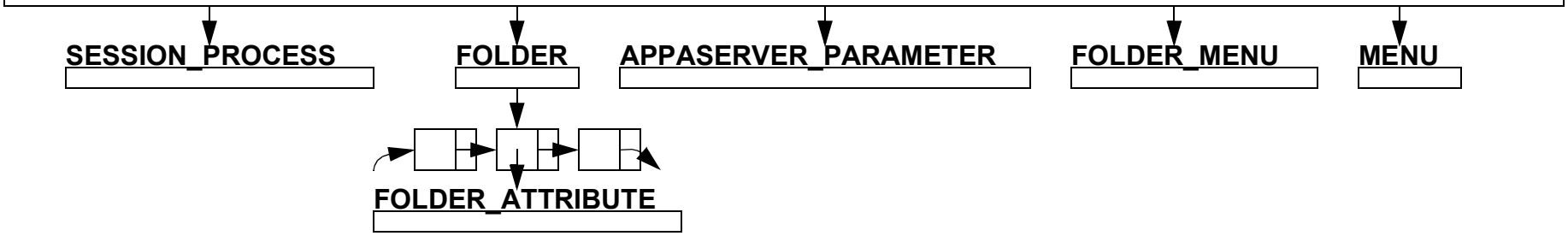
MERGE_PURGE_CHOOSE (2)

```
/* Usage */
LIST *merge_purge_choose_widget_container_list(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *process_name,
    LIST *merge_purge_choose_role_folder_name_list() );

/* Process */
LIST *widget_container_list = list_new();
int counter = 2;

list_set(
    widget_container_list,
    ( widget_container =
        WIDGET_CONTAINER *widget_container_new(
            table_open, (char *)0 ) ) );

widget_container->table_open->border_boolean = 1;
widget_container->table_open->cell_spacing = 3;
widget_container->table_open->cell_padding = 3;

for *folder_name in
    merge_purge_choose_role_folder_name_list
{
    if ( ++counter == 3 )
    {
        list_set(
            widget_container_list,
            WIDGET_CONTAINER *widget_container_new(
                table_row, (char *)0 ) );

        counter = 0;
    }

    list_set(
        widget_container_list,
        WIDGET_CONTAINER *widget_container_new(
            table_data, (char *)0 ) );

    list_set(
        widget_container_list,
        WIDGET_CONTAINER *
            merge_purge_choose_widget_container(
                application_name,
                session_key,
                login_name,
                role_name,
                process_name,
                folder_name ) );
}

list_set(
    widget_container_list,
    WIDGET_CONTAINER *widget_container_new(
        table_close, (char *)0 ) );

/* Usage */
WIDGET_CONTAINER *
    merge_purge_choose_widget_container(
        char *application_name,
        char *session_key,
        char *login_name,
        char *role_name,
        char *process_name,
        char *folder_name );

/* Process */
WIDGET_CONTAINER *widget_container =
    widget_container_new(
        (char *)0 /* widget_name */,
        anchor );

char *merge_purge_hypertext_reference(
    MERGE_PURGE_CHOOSE_POST_EXECUTABLE,
    application_name,
    session_key,
    login_name,
    role_name,
    process_name,
    folder_name );

widget_container->anchor->hypertext_reference =
    merge_purge_hyptertext_reference();

widget_container->anchor->target_frame =
    FRAMESET_PROMPT_FRAME;

char *merge_purge_choose_prompt(
    char *folder_name );

widget_container->anchor->prompt =
    merge_purge_choose_prompt();

/* Driver */
printf(   "%s\n",
            merge_purge_choose->
                document_form_html );

/* Attributes */
SESSION_PROCESS *session_process;
char *title_string;
char *sub_sub_title_string;
PROCESS *process;
char *document_body_html;
char *role_folder_where;
LIST *role_folder_name_list;
LIST *widget_container_list;
char *widget_container_list_html;
char *document_form_html;
```



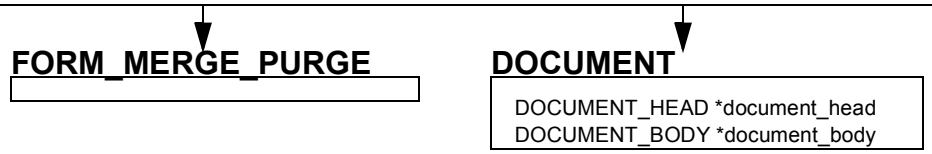
MERGE_PURGE_FOLDER (1)

```
/* Usage */
MERGE_PURGE_FOLDER *
    merge_purge_folder_new(
        int argc,
        char **argv,
        char *application_name,
        char *session_key,
        char *login_name,
        char *role_name,
        char *process_name,
        char *folder_name );

/* Process */
MERGE_PURGE_FOLDER *merge_purge_folder_calloc(
    void );

/* Sets appaserver environment and outputs argv. */
/* Each parameter is security inspected. */
/* Any error will exit( 1 ). */
SESSION_PROCESS *session_process =
    session_process_integrity_exit(
        argc,
        argv,
        application_name,
        session_key,
        login_name,
        role_name,
        process_name );

char *folder_name =
    security_sql_injection_escape(
          SECRUITY_ESCAPE_CHARACTER_STRING,
        folder_name );

LIST *role_folder_list(
    role_name,
    this->folder_name );

boolean session_folder_valid(
    APPASERVER_UPDATE_STATE,
    (char *)0 /* state2 */,
    this->folder_name,
    role_folder_list() );

if (!session_folder_valid() )
{
    session_access_failed_message_exit(
        application_name,
        login_name,
        session_current_ip_address(
            environment_remote_ip_address(
                ENVIRONMENT_REMOTE_ADDR ) ) );
}

FOLDER *folder =
    folder_fetch(
        this->folder_name,
        (LIST *)0 /* exclude_attribute_name_list */,
        /* Sets folder_attribute_primary_list */
        /* Sets folder_attribute_primary_key_list */
        /* Sets folder_attribute_name_list */
        1 /* fetch_folder_attribute_list */,
        1 /* fetch_attribute */,
        0 /* not cache_boolean */ );

boolean application_menu_horizontal_boolean(
    application_name );

APPASERVER_PARAMETER *appaserver_parameter =
    appaserver_parameter_new();

if ( application_menu_horizontal_boolean() )
{
    FOLDER_MENU *folder_menu =
        folder_menu_fetch(
            application_name,
            session_key,
            role_name,
            appaserver_parameter->data_directory,
            0 /* not folder_menu_remove_boolean */ );

    MENU *menu =
        menu_new(
            application_name,
            session_key,
            login_name,
            role_name,
            1 /* application_menu_horizontal_boolean */,
            folder_menu->count_list );
}

char *merge_purge_hypertext_reference(
        MERGE_PURGE_FOLDER_POST_EXECUTABLE,
        application_name,
        session_key,
        login_name,
        role_name,
        process_name,
        this->folder_name );
```



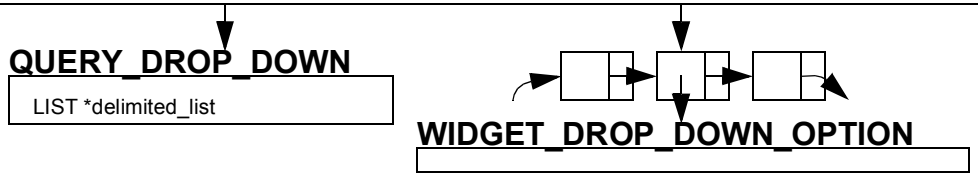
MERGE_PURGE_FOLDER (2)

```
/* Process (continued) */
char *merge_purge_folder_keep_prompt(
    char *this->folder_name );

char *merge_purge_folder_delete_prompt(
    char *this->folder_name );

FORM_MERGE_PURGE *form_merge_purge =
    form_merge_purge_new(
        MERGE_PURGE_KEEP_LABEL,
        MERGE_PURGE_DELETE_LABEL,
        application_name,
        login_name,
        this->folder_name,
        folder->no_initial_capital,
        folder->folder_attribute_primary_list,
        merge_purge_folder_keep_prompt(),
        merge_purge_folder_delete_prompt(),
        merge_purge_hypertext_reference() );

char *application_title_string( application_name );

char *merge_purge_folder_title_string(
    char *process_name,
    char *this->folder_name );

DOCUMENT *document =
    document_new(
        application_name,
        application_title_string(),
        merge_purge_folder_title_string(),
        (char *)0 /* sub_title_string */,
        (char *)0 /* sub_sub_title_string */,
        folder->notepad,
        (char *)0 /* onload_javascript_string */,
        (menu) ? menu->html : (char *)0,
        document_head_menu_setup_string(
            (menu) ? 1 : 0 /* menu_boolean */ ),
        (char *)0 /* calendar_setup_string */,
        javascript_include_string(
            application_name,
            (LIST *)0 /* javascript_filename_list */ ) );

char *document_form_html(
    document->html,
    document->document_head->html,
    document_head_close_tag(),
    document->document_body->html,
    form_merge_purge->form_html,
    document_body_close_tag(),
    document_close_tag() );

form_merge_purge_free(
    form_merge_purge );

/* Driver */
printf(   "%s\n",
          merge_purge_folder->
              document_form_html );

/* Attributes */
SESSION_PROCESS *session_process;
char *folder_name;
LIST *role_folder_list;
FOLDER *folder;
boolean application_menu_horizontal_boolean;
APPASERVER_PARAMETER *appaserver_parameter;
FOLDER_MENU *folder_menu;
MENU *menu;
char *merge_purge_hypertext_reference;
ROLE *role;
char *keep_prompt;
char *delete_prompt;
FORM_MERGE_PURGE *form_merge_purge;
char *application_title_string;
char *title_string;
DOCUMENT *document;
char *document_form_html;
```



FORM_MERGE_PURGE (1)

```
/* Usage */
FORM_MERGE_PURGE *form_merge_purge_new(
    const char *MERGE_PURGE_KEEP_LABEL,
    const char *MERGE_PURGE_DELETE_LABEL,
    char *application_name,
    char *login_name,
    char *folder_name,
    boolean folder->no_initial_capital,
    LIST *folder->folder_attribute_primary_list,
    char *merge_purge_folder_keep_prompt(),
    char *merge_purge_folder_delete_prompt(),
    char *merge_purge_hypertext_reference() );

/* Process */
FORM_MERGE_PURGE *form_merge_purge_calloc(
    void );

QUERY_DROP_DOWN *query_drop_down_new(
    application_name,
    (char *)0 /* session_key */,
    login_name,
    (char *)0 /* role_name */,
    (char *)0 /* state */,
    (char *)0 /* many_folder_name */,
    folder_name /* one_folder_name */,
    (char *)0 /* related_column */,
    folder_attribute_primary_list
        /* one_folder_attribute_list */,
    (char *)0 /* populate_drop_down_process_name */,
    (LIST *)0 /* relation_mto1_to_one_list */,
    (DICTIONARY *)0 /* drop_down_dictionary */,
    (SECURITY_ENTITY *)0 /* security_entity */,
    (LIST *)0 /* common_name_list */ );

LIST *widget_drop_down_option_list(
    ATTRIBUTE_MULTI_KEY_DELIMITER,
    WIDGET_DROP_DOWN_LABEL_DELIMITER,
    WIDGET_DROP_DOWN_EXTRA_DELIMITER,
    WIDGET_DROP_DOWN_DASH_DELIMITER,
    query_drop_down_new()->delimited_list,
    no_initial_capital );

LIST *widget_container_list = list_new();

list_set(
    widget_container_list,
    ( WIDGET_CONTAINER *widget_container =
        widget_container_new(
            table_open, (char *)0 ) ) );

widget_container->table_open->border_boolean = 1;

/* Keep row */
list_set(
    widget_container_list,
    widget_container_new(
        table_row, (char *)0 ) );

list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );

list_set(
    widget_container_list,
    widget_container_new(
        non_edit_text,
        merge_purge_folder_keep_prompt ) );

list_set(
    widget_container_list,
    widget_container_new(
        table_data, (char *)0 ) );

list_set(
    widget_container_list,
    ( keep_widget_container =
        widget_container_new(
            drop_down,
            MERGE_PURGE_KEEP_LABEL ) ) );

keep_widget_container->
    drop_down->
    widget_drop_down_option_list =
        widget_drop_down_option_list();
```

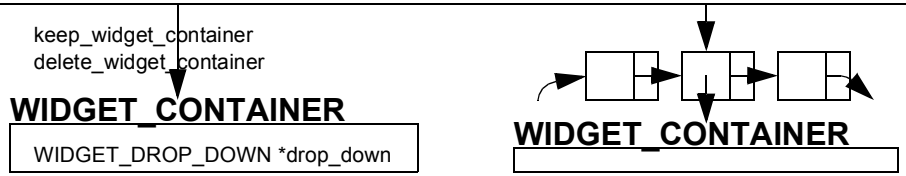


FORM_MERGE_PURGE (2)

```
/* Process (continued) */
/* Purge row */
list_set(
   widget_container_list,
   widget_container_new(
      table_row, (char *)0 ) );

list_set(
   widget_container_list,
   widget_container_new(
      table_data, (char *)0 ) );

list_set(
   widget_container_list,
   widget_container_new(
      non_edit_text,
      merge_purge_folder_delete_prompt ) );

list_set(
   widget_container_list,
   widget_container_new(
      table_data, (char *)0 ) );

list_set(
   widget_container_list,
   ( delete_widget_container =
      widget_container_new(
         drop_down,
         MERGE_PURGE_DELETE_LABEL ) ) );

delete_widget_container->
   drop_down->
   widget_drop_down_option_list =
      widget_drop_down_option_list();

/* Execute row */
list_set(
   widget_container_list,
      widget_container_new(
         table_row, (char *)0 ) );

list_set(
   widget_container_list,
   widget_container_new(
      table_data, (char *)0 ) );

list_set(
   widget_container_list,
   widget_container_new(
      non_edit_text,
      APPASERVER_EXECUTE_YN ) );

list_set(
   widget_container_list,
   widget_container_new(
      table_data, (char *)0 ) );

list_set(
   widget_container_list,
   widget_container_new(
      yes_no,
      APPASERVER_EXECUTE_YN ) );

/* Table close */
list_set(
   widget_container_list,
   widget_container_new(
      table_close, (char *)0 ) );

/* Button list row */
list_set(
   widget_container_list,
   ( WIDGET_CONTAINER *widget_container =
      widget_container_new(
         table_open, (char *)0 ) ) );

widget_container->table_open->border_boolean = 1;

list_set(
      widget_container_list,
      widget_container_new(
         table_row, (char *)0 ) );

list_set(
   widget_container_list,
   widget_container_new(
      table_data, (char *)0 ) );

list_set_list(
   widget_container_list,
   LIST *widget_button_submit_reset_container_list() );

list_set(
   widget_container_list,
   widget_container_new(
      table_close, (char *)0 ) );

char *widget_container_list_html(
   (char *)0 /* state */,
   -1 /* row_number to omit index suffix */,
   (char *)0 /* background_color */,
   widget_container_list );

char *form_tag(
   FORM_MERGE_PURGE_NAME,
   merge_purge_hypertext_reference
      /* action_string */,
   FRAMESET_TABLE_FRAME /* target_frame */ );

char *form_html(
   form_tag(),
   widget_container_list_html(),
   (char *)0 /* hidden_html */,
   form_close_tag() );
```

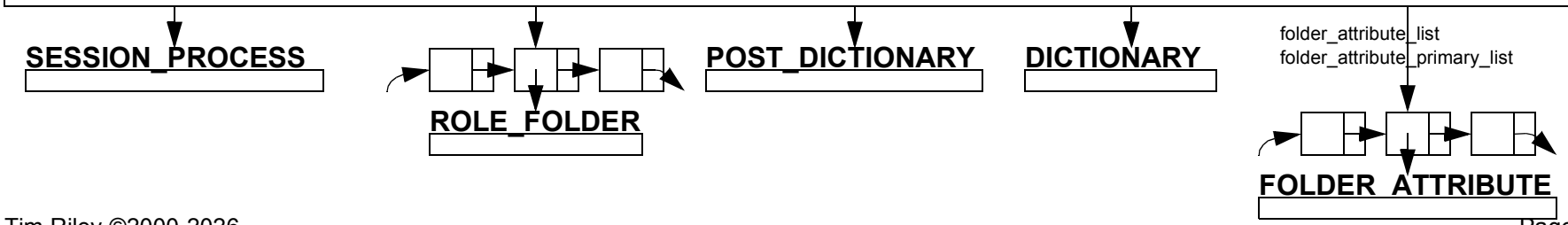
FORM_MERGE_PURGE (3)

```
/* Usage */
void form_merge_purge_free(
    FORM_MERGE_PURGE *form_merge_purge );

/* Process */
free( form_merge_purge->widget_container_list_html );
free( form_merge_purge->form_tag );
free( form_merge_purge->form_html );
free( form_merge_purge );

/* Attributes */
QUERY_DROP_DOWN *query_drop_down;
LIST *widget_drop_down_option_list;
LIST *widget_container_list;

WIDGET_CONTAINER *keep_widget_container;
WIDGET_CONTAINER *delete_widget_container;
char *widget_container_list_html;
char *form_tag;
char *form_html;
```



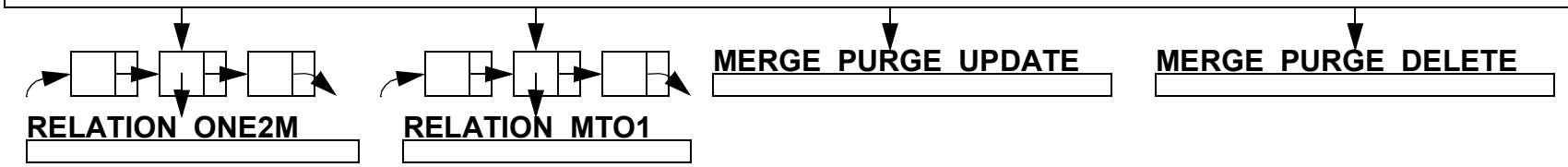
MERGE_PURGE_PROCESS (1)

```
/* Usage */
MERGE_PURGE_PROCESS *
   merge_purge_process_new(
      int argc,
      char **argv,
      char *application_name,
      char *session_key,
      char *login_name,
      char *role_name,
      char *process_name,
      char *folder_name );

/* Process */
MERGE_PURGE_PROCESS *
   merge_purge_process_calloc(
      void );

/* Sets appaserver environment and outputs argv. */
/* Each parameter is security inspected. */
/* Any error will exit( 1 ). */
SESSION_PROCESS *session_process =
   session_process_integrity_exit(
      argc,
      argv,
      application_name,
      session_key,
      login_name,
      role_name,
      process_name );

char *folder_name =
   security_sql_injection_escape(
      SECRUITY_ESCAPE_CHARACTER_STRING,
      folder_name );

LIST *role_folder_list(
   role_name,
   this->folder_name );

boolean session_folder_valid(
   APPASERVER_UPDATE_STATE,
   (char *)0 /* state2 */,
   this->folder_name,
   role_folder_list() );

if (!session_folder_valid() )
{
   session_access_failed_message_exit(
      application_name,
      login_name,
      session_current_ip_address(
         environment_remote_ip_address(
            ENVIRONMENT_REMOTE_ADDR ) ) );
}

POST_DICTIONARY *post_dictionary =
   post_dictionary_stdin_new(
      /* Used when expecting a spooled file */
      (char *)0 /* WIDGET_UPLOAD_RECALL_PREFIX */,
      (char *)0 /* application_name */,
      (char *)0 /* upload_directory */,
      (LIST *)0 /* upload_filename_list */ );

DICTIONARY *non_prefixed_dictionary =
   dictionary_separate_non_prefixed(
      DICTIONARY_SEPARATE_SORT_PREFIX,
      DICTIONARY_SEPARATE_QUERY_PREFIX,
      DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
      DICTIONARY_SEPARATE_IGNORE_PREFIX,
      DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
      DICTIONARY_SEPARATE_PAIR_PREFIX,
      post_dictionary->original_post_dictionary );

char *keep_delimited_string =
   dictionary_get(
      MERGE_PURGE_KEEP_LABEL,
      non_prefixed_dictionary );

char *delete_delimited_string =
   dictionary_get(
      MERGE_PURGE_DELETE_LABEL,
      non_prefixed_dictionary );

boolean merge_purge_process_same_boolean(
   char *keep_delimited_string,
   char *delete_delimited_string );

if ( merge_purge_process_same_boolean() )
      return this;

LIST *merge_purge_process_keep_string_list(
   char SQL_DELIMITER,
   char *keep_delimited_string );

LIST *merge_purge_process_delete_string_list(
   char SQL_DELIMITER,
   char *delete_delimited_string );

LIST *folder_attribute_list(
   this->folder_name,
   (LIST *)0 /* exclude_attribute_name_list */,
   1 /* fetch_attribute */,
   1 /* cache_boolean */ );

LIST *folder_attribute_primary_list(
   this->folder_name,
   folder_attribute_list() );

LIST *folder_attribute_primary_key_list(
   this->folder_name,
   folder_attribute_primary_list() );
```



MERGE_PURGE_PROCESS (2)

```
/* Process (continued) */
LIST *relation_one2m_list(
    this->folder_name
        /* one_folder_name */,
    folder_attribute_primary_key_list()
        /* one_folder_primary_key_list */,
    1 /* include_isa_boolean */ );

if ( !list_length( relation_one2m_list() ) exit( 1 );

LIST *relation_mto1_isa_list(
    (LIST *)0 /* mto1_isa_list Pass in null */,
    folder_name
        /* many_folder_name */,
    folder_attribute_primary_key_list()
        /* many_folder_primary_key_list */,
    0 /* not fetch_relation_one2m_list */,
    0 /* not fetch_relation_mto1_list */ );

boolean merge_purge_process_execute_boolean(
    APPASERVER_EXECUTE_YN,
    non_prefixed_dictionary );

MERGE_PURGE_UPDATE *merge_purge_update =
    merge_purge_update_new(
        application_name,
        session_key,
        login_name,
        role_name,
        this->folder_name,
        keep_string_list,
        delete_string_list,
        relation_one2m_list(),
        relation_mto1_isa_list(),
        folder_attribute_primary_list() );

if ( !merge_purge_update ) return this;

MERGE_PURGE_DELETE *merge_purge_delete =
    merge_purge_delete_new(
        application_name,
        session_key,
        login_name,
        role_name,
        this->folder_name,
        delete_delimited_string,
        folder_attribute_primary_key_list() );

if ( merge_purge_delete )
{
    char *merge_purge_process_exists_system_string(
        folder_name,
        merge_purge_delete->
            delete->
            delete_root->
            delete_where );

    boolean merge_purge_process_row_exists_boolean(
        merge_purge_process_exists_system_string() );

    free( merge_purge_process_exists_system_string() );
}

/* Attributes */
SESSION_PROCESS *session_process;
char *folder_name;
LIST *role_folder_list;
POST_DICTIONARY *post_dictionary;
DICTIONARY *non_prefixed_dictionary;
char *keep_delimited_string;
char *delete_delimited_string;
boolean same_boolean;
LIST *keep_string_list;
LIST *delete_string_list;
LIST *folder_attribute_list;
LIST *folder_attribute_primary_list;
LIST *folder_attribute_primary_key_list;
LIST *relation_one2m_list;
LIST *relation_mto1_isa_list;
boolean execute_boolean;
MERGE_PURGE_UPDATE *merge_purge_update;
MERGE_PURGE_DELETE *merge_purge_delete;
boolean row_exists_boolean;
```



MERGE_PURGE_PROCESS (3)

```
/* Usage */
boolean merge_purge_process_execute_boolean(
    const char *APPASERVER_EXECUTE_YN,
    DICTIONARY *non_prefixed_dictionary );

/* Process */
char *execute_yn =
    dictionary_get(
        APPASERVER_EXECUTE_YN,
        non_prefixed_dictionary );

return ( execute_yn && *execute_yn == 'y' );
```

```
/* Usage */
char *merge_purge_process_exists_system_string(
    char *folder_name,
    char *delete_where );

/* Process */
char *appaserver_table_name(
    folder_name );

snprintf( system_string[],
    sizeof ( system_string ),
    "select.sh \"count(1)\" %s \"%s\"",
    appaserver_table_name(),
    delete_where );
```

```
/* Usage */
boolean merge_purge_process_row_exists_boolean(
    char *merge_purge_process_exists_system_string );

/* Process */
string_pipe(
    merge_purge_process_exists_system_string );

(boolean)
(atoi( string_pipe() ) );
```

MERGE_PURGE_PROCESS (4)

```
/* Driver */
if ( !merge_purge_process->merge_purge_update )
{
    printf(
        MERGE_PURGE_NO_UPDATE_TEMPLATE,
        folder_name );
}
else
if ( merge_purge_process->same_boolean )
{
    printf( MERGE_PURGE_PROCESS_SAME_MESSAGE );
}
else
if ( !merge_purge_process->row_exists_boolean )
{
    printf(
        MERGE_PURGE_ALREADY_OCCURRED_TEMPLATE,
        folder_name );
}
else
if ( execute_boolean )
{
    void merge_purge_process_execute(
        SQL_EXECUTABLE,
        application_name,
        merge_purge_process->merge_purge_update,
        merge_purge_process->merge_purge_delete );

    void process_increment_count(
        process_name );

    printf( "<h3>Process complete</h3>\n" );
}
else
{
    void merge_purge_process_html_display(
        merge_purge_process->delete_delimited_string,
        merge_purge_process->
            merge_purge_delete->
            delete );
}
```

```
/* Usage */
void merge_purge_process_execute(
    const char *SQL_EXECUTABLE,
    char *application_name,
    MERGE_PURGE_UPDATE *merge_purge_update,
    MERGE_PURGE_DELETE *merge_purge_delete );

/* Process */
void update_row_list_zap_root(
    merge_purge_update->
        update->
        update_row_list
            /* Sets update_row->update_root = null */ );

(void)update_row_list_execute(
    SQL_EXECUTABLE,
    application_name,
    0 /* not update_root_boolean */,
    merge_purge_update->
        update->
        update_row_list,
    merge_purge_process->
        update->
        appaserver_error_filename );

void update_row_list_command_line_execute(
    merge_purge_update->
        update->
        update_row_list );

(void)delete_execute(
    SQL_EXECUTABLE,
    merge_purge_delete->
        delete->
        delete_input->
        appaserver_error_filename,
    merge_purge_delete->
        delete->
        delete_sql_list,
    merge_purge_delete->
        delete->
        pre_command_list,
    merge_purge_delete->
        delete->
        command_list );

/* Usage */
void merge_purge_process_html_display(
    char *delete_delimited_string,
    UPDATE *merge_purge_process->
        merge_purge_update->
        update,
    DELETE *merge_purge_process->
        merge_purge_delete->
        delete );

/* Process */
char *merge_purge_process_heading_html(
    const char *
        WIDGET_DROP_DOWN_DASH_DELIMITER,
    const char SQL_DELIMITER,
    char *delete_delimited_string );

printf(
    "%s\n",
    merge_purge_heading_html() );

FILE *appaserver_output_pipe(
    "html_paragraph_wrapper.e" );

void update_row_list_display(
    update->update_row_list,
    appaserver_output_pipe() /* output_stream */ );

pclose( appaserver_output_pipe() );

for DELETE_SQL *delete_sql in delete->delete_sql_list
{
    printf( "<p>%s\n", delete_sql->statement );
}
```

MERGE_PURGE_UPDATE

```
/* Usage */
MERGE_PURGE_UPDATE *
   merge_purge_update_new(
      char *application_name,
      char *session_key,
      char *login_name,
      char *role_name,
      char *folder_name,
      LIST *keep_string_list,
      LIST *delete_string_list,
      LIST *relation_one2m_list,
      LIST *relation_mto1_isa_list,
      LIST *folder_attribute_primary_list );

/* Process */
MERGE_PURGE_UPDATE *
   merge_purge_update_calloc(
      void );

DICTIONARY *multi_row_dictionary =
   merge_purge_update_dictionary(
      keep_string_list,
      folder_attribute_primary_list );

DICTIONARY *file_dictionary =
   merge_purge_update_dictionary(
      delete_string_list,
      folder_attribute_primary_list );
```

```
UPDATE *update_new(
   application_name,
   session_key,
   login_name,
   role_name,
   folder_name,
   0 /* not override_row_restrictions */,
   multi_row_dictionary,
   file_dictionary,
   (FOLDER_ROW_LEVEL_RESTRICTION *)0,
   relation_one2m_list,
   (LIST *)0 /* relation_mto1_list */,
   relation_mto1_isa_list,
   folder_attribute_primary_list
      /* folder_attribute_append_isa_list */,
   (PROCESS *)0 /* post_change_process */,
   0 /* not update_root_boolean */ );

if ( !update_new() ) return null;

/* Usage */
DICTIONARY *merge_purge_update_dictionary(
   LIST *string_list,
   LIST *folder_attribute_primary_list );

/* Process */
DICTIONARY *dictionary_small();

list_rewind( string_list );
```

```
for FOLDER_ATTRIBUTE *folder_attribute in
   folder_attribute_primary_list
{
   char *merge_purge_update_key(
      folder_attribute->attribute_name );

   dictionary_set(
      dictionary_small(),
      merge_purge_update_key(),
      list_get( string_list ) );

   list_next( string_list );
}

/* Usage */
char *merge_purge_update_key(
   char *attribute_name );

/* Process */
sprintf(
   key[],
   “%s_1”,
   attribute_name );

/* Attributes */
DICTIONARY *multi_row_dictionary;
DICTIONARY *file_dictionary;
UPDATE *update;
```

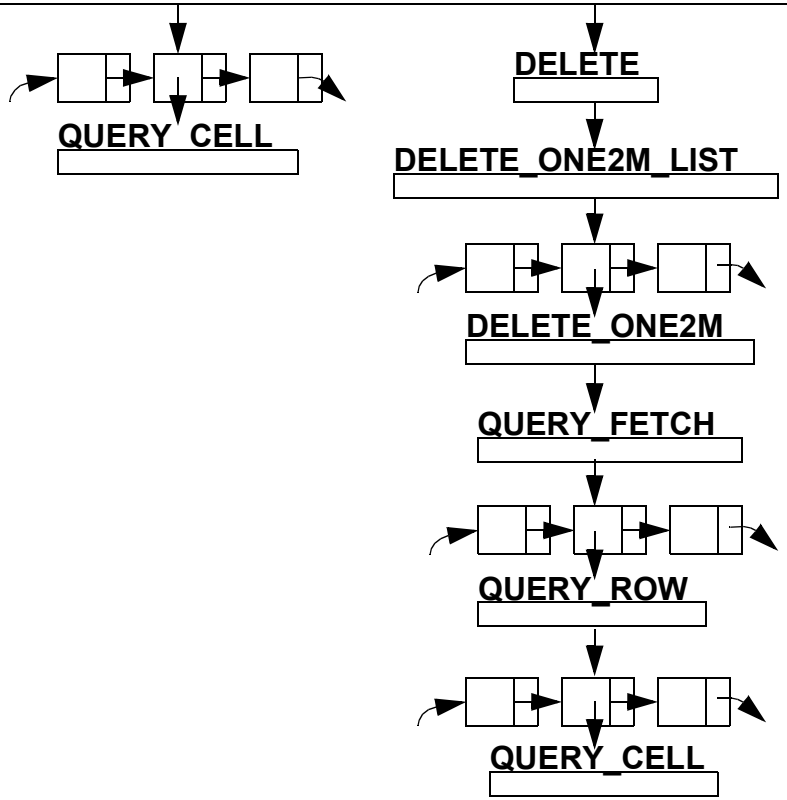
UPDATE

MERGE_PURGE_DELETE

```
/* Usage */
MERGE_PURGE_DELETE *
   merge_purge_delete_new(
      char *application_name,
      char *login_name,
      char *role_name,
      char *folder_name,
      char *delete_delimited_string,
      LIST *folder_attribute_primary_key_list() );

/* Process */
MERGE_PURGE_DELETE *
   merge_purge_delete_calloc(
      void );
```

```
LIST *delete_primary_query_cell_list(
   SQL_DELIMITER,
   delete_delimited_string,
   folder_attribute_primary_key_list );

DELETE *delete_new(
   application_name,
   login_name,
   role_name,
   folder_name,
   1 /* row_number */,
   1 /* row_count */,
   delete_primary_query_cell_list(),
   1 /* isa_boolean */,
   0 /* not update_null_boolean */ );
```

```
if ( !delete->folder_operation_delete ) return NULL;

if ( delete->delete_one2m_list )
{
   delete_one2m_update_zap(
      delete->delete_one2m_list->list
         /* delete_one2m_list */ );
}

/* Attributes */
LIST *delete_primary_query_cell_list;
DELETE *delete;
```



GENERIC_LOAD

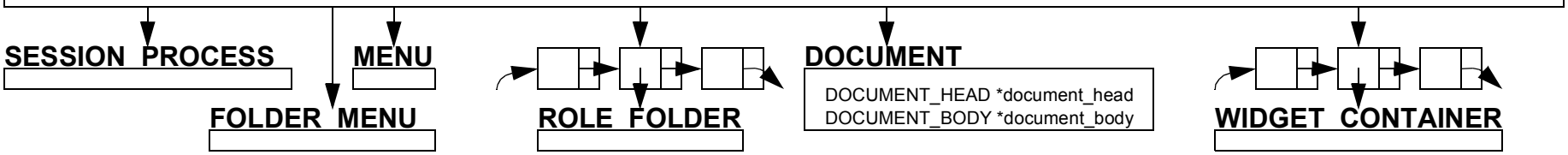
```
/* Pubiic */
char *generic_load_hypertext_reference(
    char *executable,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *process_name,
    char *folder_name );
```

GENERIC_LOAD_CHOOSE (1)

```
/* Usage */
GENERIC_LOAD_CHOOSE *generic_load_choose_new(
    int argc,
    char **argv,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *process_name );

/* Process */
GENERIC_LOAD_CHOOSE *generic_load_choose_calloc(
    void );

/* Sets appaserver environment and outputs argv. */
/* Each parameter is security inspected. */
/* Any error will exit( 1 ). */
SESSION_PROCESS *session_process =
    session_process_integrity_exit(
        argc,
        argv,
        application_name,
        session_key,
        login_name,
        role_name,
        process_name );

boolean application_menu_horizontal_boolean(
    application_name );

if ( application_menu_horizontal_boolean() )
{
    FOLDER_MENU *folder_menu =
        folder_menu_fetch(
            application_name,
            session_key,
            role_name,
            appaserver_parameter_data_directory()
            0 /* not folder_menu_remove_boolean */ );

    MENU *menu =
        menu_new(
            application_name,
            session_key,
            login_name,
            role_name,
            1 /* application_menu_horizontal_boolean */,
            folder_menu->count_list );
}

if ( menu )
{
    LIST *role_folder_insert_list =
        menu->role_folder_insert_list;
}
else
{
    LIST *role_folder_insert_list( role_name );
}

char *generic_load_choose_title_string(
    char *process_name );

DOCUMENT *document =
    document_new(
        application_name,
        application_title_string( application_name ),
        char *generic_load_choose_title_string(),
        (char *)0 /* sub_title_string */,
        (char *)0 /* sub_sub_title_string */,
        (char *)0 /* notepad */,
        (char *)0 /* onload_javascript_string */,
        (menu) ? menu->html : (char *)0,
        document_head_menu_setup_string(
            (menu) ? menu->html : (char *)0 ),
        (char *)0 /* calendar_setup_string */,
        javascript_include_string(
            application_name,
            (LIST *)0 /* javascript_filename_list */ ) );

LIST *generic_load_choose_widget_container_list(
    application_name,
    session_key,
    login_name,
    role_name,
    process_name,
    role_folder_insert_list );

char *widget_container_list_html(
    (char *)0 /* state */,
    -1 /* row_number to omit index suffix */,
    (char *)0 /* background_color */,
    widget_container_list );

char *document_form_html(
    document->html,
    document->document_head->html,
    document_head_close_tag(),
    document->document_body->html,
    widget_container_list_html(),
    document_body_close_tag(),
    document_close_tag() );

free( widget_container_list_html() );
```

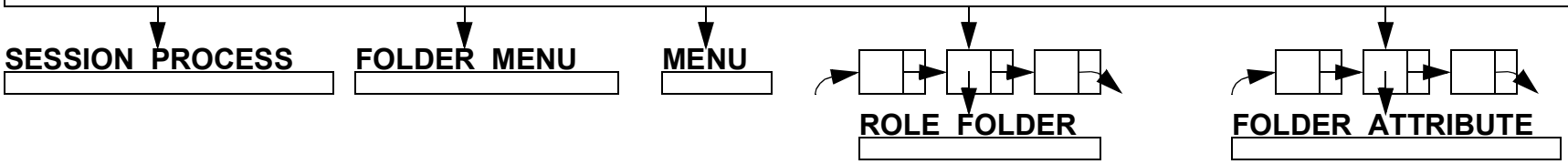
GENERIC_LOAD_CHOOSE (2)

```
/* Usage */
LIST *generic_load_choose_widget_container_list(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *process_name,
    LIST *role_folder_insert_list );

/* Process */
LIST *widget_container_list = list_new();
int counter = 2;

list_set(
    widget_container_list,
    ( widget_container =
        WIDGET_CONTAINER *widget_container_new(
            table_open, (char *)0 ) ) );

widget_container->table_open->border_boolean = 1;

for ROLE_FOLDER *role_folder in
    role_folder_insert_list
{
    if ( ++counter == 3 )
    {
        list_set(
            widget_container_list,
            WIDGET_CONTAINER *widget_container_new(
                table_row, (char *)0 ) );

        counter = 0;
    }

    list_set(
        widget_container_list,
        WIDGET_CONTAINER *widget_container_new(
            table_data, (char *)0 ) );

    list_set(
        widget_container_list,
        WIDGET_CONTAINER *
            generic_load_choose_widget_container(
                application_name,
                session_key,
                login_name,
                role_name,
                process_name,
                role_folder->folder_name ) );
}

list_set(
    widget_container_list,
    WIDGET_CONTAINER *widget_container_new(
        table_close, (char *)0 ) );

/* Usage */
WIDGET_CONTAINER *
    generic_load_choose_widget_container(
        char *application_name,
        char *session_key,
        char *login_name,
        char *role_name,
        char *process_name,
        char *folder_name );

/* Process */
WIDGET_CONTAINER *widget_container =
    widget_container_new(
        (char *)0 /* widget_name */,
        anchor );

char *generic_load_hypertext_reference(
    GENERIC_LOAD_FOLDER_EXECUTABLE,
    application_name,
    session_key,
    login_name,
    role_name,
    process_name,
    folder_name );

widget_container->anchor->hypertext_reference =
    generic_load_hyptertext_reference();

widget_container->anchor->target_frame =
    FRAMESET_PROMPT_FRAME;

char *generic_load_choose_prompt(
    char *folder_name );

widget_container->anchor->prompt =
    generic_load_choose_prompt();

/* Driver */
printf(   "%s\n",
          generic_load_choose->
              document_form_html );

/* Attributes */
SESSION_PROCESS *session_process;
boolean application_menu_horizontal_boolean;
FOLDER_MENU *folder_menu;
MENU *menu;
LIST *role_folder_insert_list;
char *title_string;
DOCUMENT *document;
LIST *widget_container_list;
char *widget_container_list_html;
char *document_form_html;
```



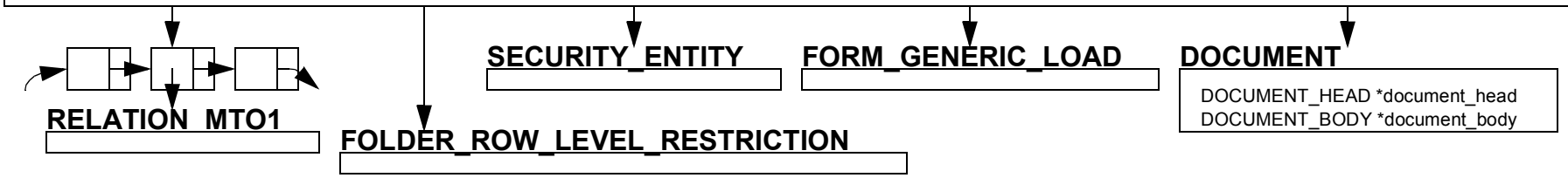
GENERIC_LOAD_FOLDER (1)

```
/* Usage */
GENERIC_LOAD_FOLDER *generic_load_folder_new(
    int argc,
    char **argv,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *process_name,
    char *folder_name );

/* Process */
GENERIC_LOAD_FOLDER *generic_load_folder_calloc(
    void );

/* Sets appaserver environment and outputs argv. */
/* Each parameter is security inspected. */
/* Any error will exit( 1 ). */
SESSION_PROCESS *session_process =
    session_process_integrity_exit(
        argc,
        argv,
        application_name,
        session_key,
        login_name,
        role_name,
        process_name );

char *folder_name =
    security_sql_injection_escape(
        SECRUITY_ESCAPE_CHARACTER_STRING,
        folder_name );

boolean application_menu_horizontal_boolean(
    application_name );

APPASERVER_PARAMETER *appaserver_parameter =
    appaserver_parameter_new();

if ( application_menu_horizontal_boolean() )
{
    FOLDER_MENU *folder_menu =
        folder_menu_fetch(
            application_name,
            session_key,
            role_name,
            appaserver_parameter->data_directory,
            0 /* not folder_menu_remove_boolean */ );

    MENU *menu =
        menu_new(
            application_name,
            session_key,
            login_name,
            role_name,
            1 /* application_menu_horizontal_boolean */,
            folder_menu->count_list );
}

if ( menu )
{
    LIST *role_folder_insert_list =
        menu->role_folder_insert_list;
}
else
{
    LIST *role_folder_insert_list( role_name );
}

if ( !role_folder_state_boolean(
        APPASERVER_INSERT_STATE,
        folder_name,
        role_folder_insert_list ) )
{
    exit( 1 );
}

char *generic_load_hypertext_reference(
        GENERIC_LOAD_INSERT_EXECUTABLE,
        application_name,
        session_key,
        login_name,
        role_name,
        process_name,
        folder_name );

ROLE *role =
    role_fetch(
        role_name,
        1 /* fetch_role_attribute_exclude_list */ );

LIST *role_attribute_exclude_insert_name_list(
    APPASERVER_INSERT_STATE,
    role->role_attribute_exclude_list );

LIST *folder_attribute_list(
    folder_name,
    role_attribute_exclude_insert_name_list(),
    1 /* fetch_attribute */,
    1 /* cache_boolean */ );

LIST *folder_attribute_primary_key_list(
    folder_name,
    folder_attribute_list );

LIST *relation_mto1_recursive_list(
    (LIST *)0 /* relation_mto1_list Pass in null */,
    role_name,
    folder_name
        /* many_folder_name */,
    folder_attribute_primary_key_list()
        /* many_folder_primary_key_list */,
    1 /* single_foreign_key_only */ );

FOLDER_ROW_LEVEL_RESTRICTION *
    folder_row_level_restriction =
        folder_row_level_restriction_fetch(
            folder_name );

SECURITY_ENTITY *security_entity =
    security_entity_new(
        login_name,
        folder_name,
        folder_row_level_restriction->non_owner_forbid,
        role->override_row_restrictions );
```



GENERIC_LOAD_FOLDER (2)

```
/* Process (continued) */
FORM_GENERIC_LOAD *form_generic_load =
    form_generic_load_new(
        GENERIC_LOAD_FILENAME_LABEL,
        GENERIC_LOAD_SKIP_HEADER_ROWS_LABEL,
        GENERIC_LOAD_EXECUTE_YN_LABEL,
        GENERIC_LOAD_POSITION_PREFIX,
        GENERIC_LOAD_CONSTANT_PREFIX,
        GENERIC_LOAD_IGNORE_PREFIX,
        application_name,
        login_name,
        role_name,
        folder_name,
        generic_load_hypertext_reference(),
        folder_attribute_list(),
        relation_mto1_recursive_list(),
        security_entity );

char *application_title_string( application_name );

char *generic_load_folder_title_string(
    char *application_title_string(),
    char *folder_name );

LIST *generic_load_folder_javascript_filename_list(
    void );

char *javascript_include_string(
    application_name,
    generic_load_folder_javascript_filename_list() );

int folder_attribute_calendar_date_name_list_length(
    folder_attribute_list );

char *document_head_calendar_setup_string(
    folder_attribute_calendar_date_name_list_length() );

char *folder_notepad( folder_name );

DOCUMENT *document =
    document_new(
        application_name,
        application_title_string(),
        generic_load_folder_title_string(),
        (char *)0 /* sub_title_string */,
        (char *)0 /* sub_sub_title_string */,
        folder_notepad(),
        (char *)0 /* onload_javascript_string */,
        (menu) ? menu->html : (char *)0,
        document_head_menu_setup_string(
            (menu) ? 1 : 0 /* menu_boolean */ ),
        document_head_calendar_setup_string(),
        javascript_include_string() );

char *document_form_html(
    document->html,
    document->document_head->html,
    document_head_close_tag(),
    document->document_body->html,
    form_generic_load->form_html,
    document_body_close_tag(),
    document_close_tag() );

free(
    form_generic_load->
        widget_container_list_html );

free(
    form_generic_load->
        form_html );

/* Driver */
printf(   "%s\n",
          generic_load_folder->
              document_form_html );

/* Attributes */
SESSION_PROCESS *session_process;
char *folder_name;
boolean application_menu_horizontal_boolean;
APPASERVER_PARAMETER *appaserver_parameter;
FOLDER_MENU *folder_menu;
MENU *menu;
LIST *role_folder_insert_list;
char *generic_load_hypertext_reference;
ROLE *role;
LIST *role_attribute_exclude_insert_name_list;
LIST *folder_attribute_list;
LIST *folder_attribute_primary_key_list;
LIST *relation_mto1_recursive_list;
FOLDER_ROW_LEVEL_RESTRICTION *
    folder_row_level_restriction;
SECURITY_ENTITY *security_entity;
FORM_GENERIC_LOAD *form_generic_load;
char *application_title_string;
char *title_string;
LIST *javascript_module_list;
char *javascript_include_string;
int folder_attribute_calendar_date_name_list_length;
char *document_head_calendar_setup_string;
char *folder_notepad;
DOCUMENT *document;
char *document_form_html;
```



FORM_GENERIC_LOAD (1)

```
/* Usage */
FORM_GENERIC_LOAD *form_generic_load_new(
    char *GENERIC_LOAD_FILENAME_LABEL,
    char *GENERIC_LOAD_SKIP_HEADER_ROWS_LABEL,
    char *GENERIC_LOAD_EXECUTE_YN_LABEL,
    char *GENERIC_LOAD_POSITION_PREFIX,
    char *GENERIC_LOAD_CONSTANT_PREFIX,
    char *GENERIC_LOAD_IGNORE_PREFIX,
    char *application_name,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *generic_load_hypertext_reference(),
    LIST *folder_attribute_list(),
    LIST *relation_mto1_recursive_list(),
    SECURITY_ENTITY *security_entity );

/* Process */
FORM_GENERIC_LOAD *form_generic_load_calloc(
    void );

char *form_generic_load_action_string(
    char *generic_load_hypertext_reference );

char *form_tag(
        FORM_GENERIC_LOAD_FORM_NAME,
        form_generic_load_action_string(),
        FRAMESET_TABLE_FRAME );

LIST *widget_container_list = list_new();

list_set_list(
    widget_container_list,
    form_generic_load_button_container_list() );

list_set_list(
    widget_container_list,
    form_generic_load_upload_container_list(
        GENERIC_LOAD_FILENAME_LABEL,
        GENERIC_LOAD_SKIP_HEADER_ROWS_LABEL,
        GENERIC_LOAD_EXECUTE_YN_LABEL ) );

list_set(
    widget_container_list,
    widget_container_new(
        line_break, (char *)0 ) );

list_set(
    widget_container_list,
    ( WIDGET_CONTAINER *widget_container =
        widget_container_new(
            table_open, (char *)0 ) ) );

widget_container->table_open->border_boolean = 1;

list_set_list(
    widget_container_list,
    form_generic_load_heading_container_list() );

int default_position = 0;

for FOLDER_ATTRIBUTE *folder_attribute in
    folder_attribute_list
{
    ++default_position;

    list_set(
        widget_container_list,
        widget_container_new(
            table_row, (char *)0 ) );

    list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );

    char *form_generic_load_attribute_prompt(
        char *folder_attribute->prompt,
        char *folder_attribute->attribute->hint_message );

    list_set(
        widget_container_list,
        widget_container_new(
            character,
            form_generic_load_attribute_prompt() ) );

char *form_generic_load_position_label(
    char *GENERIC_LOAD_POSITION_PREFIX,
    char *folder_attribute->attribute_name );

list_set(
    widget_container_list,
    ( widget_container = widget_container_new(
        character,
        form_generic_load_position_label() ) );

widget_container->
    text->
    attribute_width_max_length = 3;

char *form_generic_load_constant_label(
    char *GENERIC_LOAD_CONSTANT_PREFIX,
    char *folder_attribute->attribute_name );

int form_generic_load_default_position(
    int folder_attribute->primary_key_index,
    int default_position );

char *form_generic_load_after_field_onchange(
    form_generic_load_position_label(),
    form_generic_load_constant_label(),
    form_generic_load_default_position() );

widget_container->
    text->
    post_change_javascript =
        form_generic_load_after_field_onchange();

widget_container->
    text->
    value_string =
        strdup(
            string_itoa(
                default_position ) );

list_set(
    widget_container_list,
    widget_container_new(
        table_data, (char *)0 );
```

FORM_GENERIC_LOAD (2)

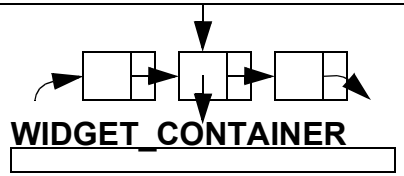
```
/* Process (continued) */
   if (    list_length( relation_mto1_recursive_list )
   &&   ( form_generic_load_relation =
            FORM_GENERIC_LOAD_RELATION *
            form_generic_load_relation_new(
               application_name,
               login_name,
               role_name,
               relation_mto1_recursive_list,
               security_entity,
               folder_attribute->attribute_name,
               form_generic_load_constant_label(),
               form_generic_load_after_field_onchange() ) ) )
   {
      list_set(
         widget_container_list,
         form_generic_load_relation->
            widget_container );
   }
   else
   {
      form_generic_load_attribute =
      FORM_GENERIC_LOAD_ATTRIBUTE *
         form_generic_load_attribute_new(
            application_name,
            login_name,
            folder_attribute->attribute_name,
            folder_attribute->attribute->datatype_name,
            folder_attribute->attribute->width,
            form_generic_load_constant_label(),
            form_generic_load_after_field_onchange() );

      list_set(
         widget_container_list,
         form_generic_load_attribute->
            widget_container );
   }

   if ( !folder_attribute->primary_key_index )
   {
      list_set(
         widget_container_list,
         widget_container_new(
            table_data, (char *)0 ) );

      char *form_generic_load_after_ignore_onchange(
         form_generic_load_position_label(),
         form_generic_load_constant_label() );

      list_set(
         widget_container_list,
         WIDGET_CONTAINER *
         form_generic_load_ignore_container(
            GENERIC_LOAD_IGNORE_PREFIX,
            folder_attribute->attribute_name,
            form_generic_load_after_ignore_onchange() );
   }
}

list_set(
   widget_container_list,
   widget_container_new(
      table_close, (char *)0 ) );

char *widget_container_list_html(
   (char *)0 /* state */,
   0 /* row_number */,
   (char *)0 /* background_color */,
   widget_container_list );

char *form_html(
   form_tag(),
   widget_container_list_html(),
   (char *)0 /* dictionary_hidden_html */,
      form_close_tag() );

free( widget_container_list_html() );

/* Usage */
LIST *form_generic_load_button_container_list( void );

/* Process */
return
LIST *widget_button_submit_reset_container_list();

/* Usage */
LIST *form_generic_load_heading_container_list(
   void );

/* Process */
LIST *widget_container_list = list_new();

list_set(
   widget_container_list,
   widget_container_new(
      table_row, (char *)0 ) );

list_set(
   widget_container_list,
   widget_container_new(
      table_heading, "Attribute" ) );

list_set(
   widget_container_list,
   widget_container_new(
      table_heading, "Position" ) );

list_set(
   widget_container_list,
   widget_container_new(
      table_heading, "Constant" ) );
```

FORM_GENERIC_LOAD (3)

```
/* Usage */
LIST *form_generic_load_upload_container_list(
    char *GENERIC_LOAD_FILENAME_LABEL,
    char *GENERIC_LOAD_SKIP_HEADER_ROWS_LABEL,
    char *GENERIC_LOAD_EXECUTE_YN_LABEL );

/* Process */
LIST *widget_container_list = list_new();

list_set(
    widget_container_list,
    ( WIDGET_CONTAINER *widget_container =
        widget_container_new(
            table_open, (char *)0 ) ) );

widget_container->table_open->border_boolean = 1;

list_set(
    widget_container_list,
    widget_container_new(
        table_row, (char *)0 ) );

list_set(
    widget_container_list,
    widget_container_new(
        table_data, (char *)0 ) );

list_set(
    widget_container_list,
    widget_container_new(
        non_edit_text, "Filename" ) );

list_set(
    widget_container_list,
    widget_container_new(
        table_data, (char *)0 ) );

list_set(
    widget_container_list,
    widget_container_new(
        upload,
        GENERIC_LOAD_FILENAME_LABEL ) );

list_set(
    widget_container_list,
    widget_container_new(
        table_row, (char *)0 ) );

list_set(
    widget_container_list,
    widget_container_new(
        table_data, (char *)0 ) );

list_set(
    widget_container_list,
    widget_container_new(
        non_edit_text, "Skip Header Rows" ) );

list_set(
    widget_container_list,
    widget_container_new(
        table_data, (char *)0 ) );

list_set(
    widget_container_list,
    ( widget_container =
        widget_container_new(
            character,
            GENERIC_LOAD_SKIP_HEADER_ROWS_LABEL )));

widget_container->
    text->
    attribute_width_max_length = 3;

list_set(
    widget_container_list,
    widget_container_new(
        table_data, (char *)0 ) );

list_set(
    widget_container_list,
    widget_container_new(
        non_edit_text,
        "How many header rows to skip?" ) );

list_set(
    widget_container_list,
    widget_container_new(
        table_row, (char *)0 ) );

list_set(
    widget_container_list,
    widget_container_new(
        table_data, (char *)0 ) );

list_set(
    widget_container_list,
    widget_container_new(
        non_edit_text, "Execute yn" ) );

list_set(
    widget_container_list,
    widget_container_new(
        yes_no,
        GENERIC_LOAD_EXECUTE_YN_LABEL ) );

list_set(
    widget_container_list,
    widget_container_new(
        table_close, (char *)0 ) );
```

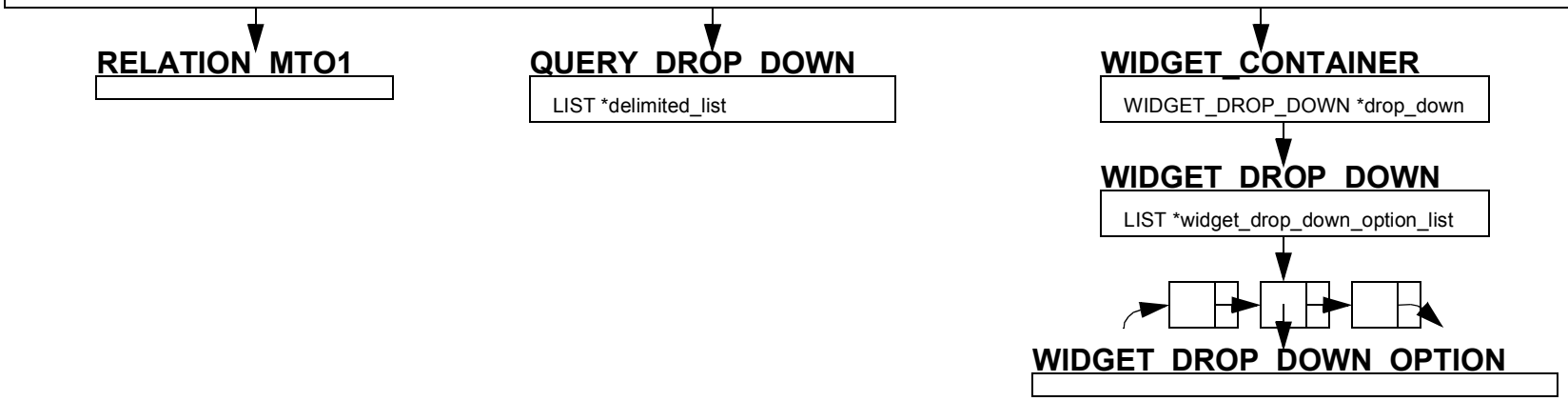
FORM_GENERIC_LOAD (4)

```
/* Usage */
WIDGET_CONTAINER *
    form_generic_load_ignore_container(
        char *GENERIC_LOAD_IGNORE_PREFIX,
        char *folder_attribute->attribute_name,
        char *
            form_generic_load_after_ignore_onchange() );

/* Process */
char *form_generic_load_ignore_label(
    char *GENERIC_LOAD_IGNORE_PREFIX,
    char *attribute_name );

WIDGET_CONTAINER *widget_container =
    widget_container_new(
        checkbox,
        form_generic_load_ignore_label() );

widget_container->
    checkbox->
    onclick =
        form_generic_load_after_ignore_onchange;

widget_container->
    checkbox->
    prompt = "Ignore";

/* Usage */
char *form_generic_load_after_field_onchange(
    char *form_generic_load_position_label(),
    char *form_generic_load_constant_label(),
    int form_generic_load_default_position() );

/* Process */
sprintf(    char onchange[],
            "after_field( '%s', '%s', %d )",
            form_generic_load_position_label,
            form_generic_load_constant_label,
            form_generic_load_default_position );

char *strdup( onchange );

/* Usage */
char *form_generic_load_after_ignore_onchange(
    char *form_generic_load_position_label(),
    char *form_generic_load_constant_label() );

/* Process */
sprintf(    char onchange[],
            "after_ignore( this, '%s', '%s' )",
            form_generic_load_position_label,
            form_generic_load_constant_label );

char *strdup( onchange );

/* Attributes */
char *action_string;
char *form_tag;
LIST *widget_container_list;
char *widget_container_list_html /* freed */;
char *form_html;
```



FORM_GENERIC_LOAD_RELATION

```
/* Usage */
FORM_GENERIC_LOAD_RELATION *
   form_generic_load_relation_new(
      char *application_name,
      char *login_name,
      char *role_name,
      char *folder_name,
      LIST *relation_mto1_recursive_list,
      SECURITY_ENTITY *security_entity,
      char *folder_attribute->attribute_name,
      char *form_generic_load_constant_label(),
      char *form_generic_load_after_field_onchange() );

/* Process */
FORM_GENERIC_LOAD_RELATION *
   form_generic_load_relation_calloc(
      void );

RELATION_MTO1 *relation_mto1 =
   relation_mto1_consumes(
      attribute_name,
      relation_mto1_recursive_list );

if ( !relation_mto1 ) return NULL;

LIST *relation_mto1_to_one_list =
   relation_mto1_list(
      relation_mto1->
         one_folder_name
            /* many_folder_name */,
      relation_mto1->
         one_folder->
         folder_attribute_primary_key_list
            /* many_folder_primary_key_list */ );

QUERY_DROP_DOWN *query_drop_down_new(
   application_name,
   (char *)0 /* session_key */,
   login_name,
   (char *)0 /* role_name */,
   (char *)0 /* state */,
   folder_name /* many_folder_name */,
   relation_mto1->one_folder_name,
   (char *)0 /* related_column */,
   relation_mto1->
      one_folder->
      folder_attribute_list,
   relation_mto1->
      one_folder->
      populate_drop_down_process_name,
   relation_mto1_to_one_list,
   (DICTIONARY *)0 /* drop_down_dictionary */,
   security_entity,
   (LIST *common_name_list */ );

widget_container =
   widget_container_new(
      drop_down,
         form_generic_load_constant_label
            /* widget_name */ );

widget_container->
   drop_down->
   widget_drop_down_option_list =
      LIST *widget_drop_down_option_list(
         ATTRIBUTE_MULTI_KEY_DELIMITER,
         WIDGET_DROP_DOWN_LABEL_DELIMITER,
         WIDGET_DROP_DOWN_EXTRA_DELIMITER,
         WIDGET_DROP_DOWN_DASH_DELIMITER,
         query_drop_down_new()->delimited_list,
         relation_mto1->
            one_folder->
            no_initial_capital );

widget_container->
   drop_down->
   top_select_boolean = 1;

widget_container->
   drop_down->
   post_change_javascript =
      form_generic_load_after_field_onchange;

/* Attributes */
RELATION_MTO1 *relation_mto1;
QUERY_DROP_DOWN *query_drop_down;
WIDGET_CONTAINER *widget_container;
```



FORM_GENERIC_LOAD_ATTRIBUTE

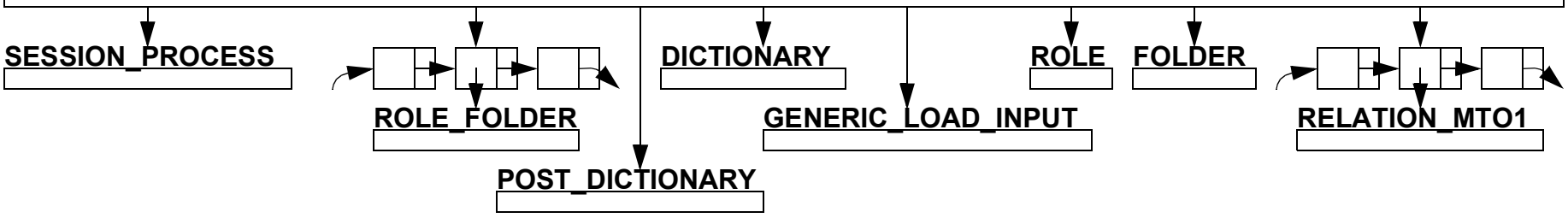
```
/* Usage */
FORM_GENERIC_LOAD_ATTRIBUTE *
    form_generic_load_attribute_new(
        char *application_name,
        char *login_name,
        char *attribute_name,
        char *datatype_name,
        int attribute_width,
        char *form_generic_load_constant_label(),
        char *form_generic_load_after_field_onchange() );

/* Process */
FORM_GENERIC_LOAD_ATTRIBUTE *
    form_generic_load_attribute_calloc(
        void );

if ( attribute_is_yes_no( attribute_name ) )
{
    widget_container =
        widget_container_new(
            yes_no,
            form_generic_load_constant_label
                /* widget_name */ );

    widget_container->
        yes_no->
            post_change_javascript =
                form_generic_load_after_field_onchange;
}
else
if ( attribute_is_date( datatype_name ) )
{
    widget_container =
        widget_container_new(
            widget_date,
            form_generic_load_constant_label
                /* widget_name */ );

    widget_container->
        date->
            application_name = application_name;

    widget_container->
        date->
        login_name = login_name;

    widget_container->
        date->
        post_change_javascript =
            form_generic_load_after_field_onchange;
}
else
{
    widget_container =
        widget_container_new(
            character,
            form_generic_load_constant_label
                /* widget_name */ ) );

    int widget_text_display_size(
        WIDGET_TEXT_DEFAULT_MAX_LENGTH,
        attribute_width );

    widget_container->
        text->
        widget_text_display_size =
            widget_text_display_size();

    widget_container->
        text->
        post_change_javascript =
            form_generic_load_after_field_onchange;
}

/* Attribute */
WIDGET_CONTAINER *widget_container;
```

WIDGET_CONTAINER

```
WIDGET_TEXT *text
WIDGET_YES_NO *yes_no
WIDGET_DATE *date
```

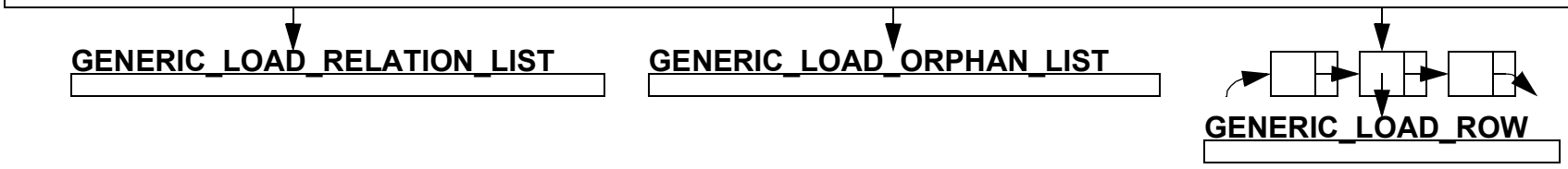
GENERIC_LOAD_INSERT (1)

```
/* Usage */
GENERIC_LOAD_INSERT *
   generic_load_insert_new(
      int argc,
      char **argv,
      char *application_name,
      char *session_key,
      char *login_name,
      char *role_name,
      char *process_name,
      char *folder_name );

/* Process */
GENERIC_LOAD_INSERT *generic_load_insert_calloc(
   void );

/* Sets appaserver environment and outputs argv. */
/* Each parameter is security inspected. */
/* Any error will exit( 1 ). */
SESSION_PROCESS *session_process =
   session_process_integrity_exit(
      argc,
      argv,
      application_name,
      session_key,
      login_name,
      role_name,
      process_name );

char *folder_name =
   char *security_sql_injection_escape(
      SECRUITY_ESCAPE_CHARACTER_STRING,
      folder_name );

LIST *role_folder_list(
   session_process->role_name,
   folder_name );

boolean session_folder_valid(
   APPASERVER_INSERT_STATE,
   (char *)0 /* state2 */,
   folder_name,
   role_folder_list() );

if (!session_folder_valid() )
{
   session_access_failed_message_exit(
      application_name,
      login_name,
      session_current_ip_address(
         environment_remote_ip_address(
            ENVIRONMENT_REMOTE_KEY ) ) );
}

char *generic_load_insert_sub_title_string(
   char *folder_name );

char *appaserver_parameter_upload_directory();

LIST *generic_load_insert_upload_filename_list(
   void );

POST_DICTIONARY *post_dictionary =
   post_dictionary_stdin_new(
      /* Used when expecting a spooled file */
      WIDGET_UPLOAD_RECALL_PREFIX,
      application_name,
      appaserver_parameter_upload_directory(),
      generic_load_insert_upload_filename_list() );

DICTIONARY *non_prefixed_dictionary =
   dictionary_separate_non_prefixed(
      DICTIONARY_SEPARATE_SORT_PREFIX,
      DICTIONARY_SEPARATE_QUERY_PREFIX,
      DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
      DICTIONARY_SEPARATE_IGNORE_PREFIX,
      DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
      DICTIONARY_SEPARATE_PAIR_PREFIX,
      post_dictionary->original_post_dictionary );

GENERIC_LOAD_INPUT *
   generic_load_input =
      generic_load_input_new(
         GENERIC_LOAD_FILENAME_LABEL,
         GENERIC_LOAD_SKIP_HEADER_ROWS_LABEL,
         GENERIC_LOAD_EXECUTE_YN_LABEL,
         GENERIC_LOAD_POSITION_PREFIX,
         GENERIC_LOAD_CONSTANT_PREFIX,
         non_prefixed_dictionary );

if ( !generic_load_input ) return NULL;

ROLE *role =
   role_fetch(
      session_process->role_name,
      1 /* fetch_role_attribute_exclude_list */ );

LIST *role_attribute_exclude_insert_name_list(
   APPASERVER_INSERT_STATE,
   role->role_attribute_exclude_list );

FOLDER *folder =
   folder_fetch(
      folder_name,
      role_attribute_exclude_insert_name_list(),
      /* Sets folder_attribute_primary_list */
      /* Sets folder_attribute_primary_key_list */
      /* Sets folder_attribute_name_list */
      1 /* fetch_folder_attribute_list */,
      1 /* fetch_attribute */,
      1 /* cache_boolean */ );

LIST *relation_mto1_list(
   folder_name
      /* many_folder_name */,
   folder->folder_attribute_primary_key_list
      /* many_folder_primary_key_list */ );
```



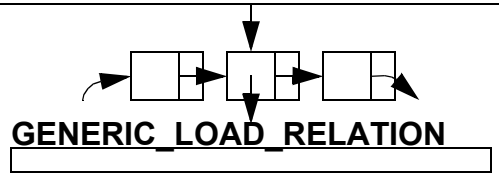
GENERIC_LOAD_INSERT (2)

```
/* Process (continued) */
char *appaserver_table_name(
    folder_name );

if ( list_length( relation_mto1_list() ) )
{
    GENERIC_LOAD_RELATION_LIST *
        generic_load_relation_list =
            generic_load_relation_list_new(
                application_name,
                relation_mto1_list() );

    generic_load_relation_list_set(
        generic_load_relation_list /* in/out */,
            generic_load_input->
                load_filename,
            generic_load_input->
                skip_header_rows_integer,
            generic_load_input->
                position_dictionary;
            generic_load_input->
                constant_dictionary,
            folder->folder_attribute_list );

    GENERIC_LOAD_ORPHAN_LIST *
        generic_load_orphan_list =
            generic_load_orphan_list_new(
                generic_load_relation_list );

    if ( generic_load_orphan_list ) return this;
}

LIST *generic_load_row_list(
    generic_load_input->
        load_filename,
    generic_load_input->
        skip_header_rows_integer,
    generic_load_input->
        position_dictionary;
    generic_load_input->
        constant_dictionary,
    folder->folder_attribute_list,
    appaserver_table_name() );

char *generic_load_insert_system_string(
    void );

/* Driver */
document_process_output(
    generic_load_insert->
        session_process->
        application_name,
    (LIST *)0 /* javascript_filename_list */,
    generic_load_insert->
        session_process->
        process_name
        /* title_string */ );

printf(
    "<h2>%s</h2>\n",
    generic_load_insert->
        generic_load_insert->sub_title_string );

if ( !generic_load_insert )
{
    printf( "<h3>No rows loaded.</h3>\n" );
}
else
{
    if ( generic_load_insert->generic_load_orphan_list )
    {
        generic_load_orphan_list_output(
            generic_load_insert->generic_load_orphan_list );
    }
    else
    if ( generic_load_insert->
            generic_load_input->
            execute )
    {
        void generic_load_row_insert(
            generic_load_insert->generic_load_row_list,
            generic_load_insert->system_string );
    }
    else
    {
        void generic_load_row_display(
            folder_name,
            generic_load_insert->generic_load_row_list );
    }
}

/* Attributes */
SESSION_PROCESS *session_process;
char *folder_name;
LIST *role_folder_list;
char *sub_title_string;
LIST *upload_filename_list;
POST_DICTIONARY *post_dictionary;
DICTIONARY *non_prefixed_dictionary;
GENERIC_LOAD_INPUT *generic_load_input;
ROLE *role;
LIST *role_attribute_exclude_insert_name_list;
FOLDER *folder;
LIST *relation_mto1_list;
GENERIC_LOAD_RELATION_LIST *
    generic_load_relation_list;
GENERIC_LOAD_ORPHAN_LIST *
    generic_load_orphan_list;
char *appaserver_table_name;
LIST *generic_load_row_list;
char *system_string;
```



GENERIC_LOAD_RELATION_LIST

```
/* Usage */
GENERIC_LOAD_RELATION_LIST *
   generic_load_relation_list_new(
      char *application_name,
      LIST *relation_mto1_list );

/* Process */
GENERIC_LOAD_RELATION_LIST *
   generic_load_relation_list_calloc(
      void );

LIST *list = list_new();

for RELATION_MTO1 *relation_mto1 in
   relation_mto1_list
{
   list_set(
      list,
      GENERIC_LOAD_RELATION *
         generic_load_relation_new(
            application_name,
            relation_mto1->one_folder_name,
            relation_mto1->relation_foreign_key_list ) );
}

/* Usage */
void generic_load_relation_list_set(
   GENERIC_LOAD_RELATION_LIST *
      generic_load_relation_list /* in/out */,
   char *generic_load_input->load_filename,
   int skip_header_rows_integer,
   DICTIONARY *position_dictionary,
   DICTIONARY *constant_dictionary,
   LIST *folder_attribute_list );

/* Process */
int row_number = 0;

FILE *generic_load_row_input_file(
   char *load_filename );

for char input[] in string_input(
   generic_load_row_input_file() )
{
   if ( ++row_number <= skip_header_rows_integer )
      continue;

   for GENERIC_LOAD_RELATION *
      generic_load_relation in
         generic_load_relation_list->list
   {
      void generic_load_relation_set(
         generic_load_relation /* in/out */,
         position_dictionary,
         constant_dictionary,
         folder_attribute_list,
         input );
   }
}

fclose( generic_load_row_input_file() );

/* Attribute */
LIST *list;
```

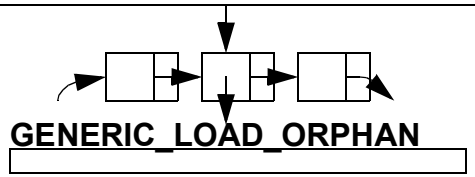


GENERIC_LOAD_RELATION

```
/* Usage */
GENERIC_LOAD_RELATION *
    generic_load_relation_new(
        char *application_name,
        char *one_folder_name,
        LIST *relation_foreign_key_list );

/* Process */
GENERIC_LOAD_RELATION *
    generic_load_relation_calloc(
        void );

char *appaserver_table_name(
    one_folder_name );

LIST *foreign_delimited_list = list_new();

/* Usage */
void generic_load_relation_set(
    GENERIC_LOAD_RELATION *
        generic_load_relation /* in/out */,
    DICTIONARY *position_dictionary,
    DICTIONARY *constant_dictionary,
    LIST *folder_attribute_list,
    char *input );

/* Process */
LIST *foreign_data_list = list_new();

LIST *generic_load_attribute_list(
    position_dictionary,
    constant_dictionary,
    folder_attribute_list,
    input );

for char *foreign_key in
    generic_load_relation->
        relation_foreign_key_list
{
    GENERIC_LOAD_ATTRIBUTE *
        generic_load_attribute =
            generic_load_attribute_seek(
                generic_load_attribute_list(),
                foreign_key /* attribute_name */ );

    if ( generic_load_attribute )
    {
        list_set(
            foreign_data_list,
            generic_load_attribute->value );
    }
}

if ( !list_length( foreign_data_list ) ) return;

char *generic_load_relation_data_delimited(
    char SQL_DELIMITER,
    LIST *foreign_data_list );

if ( list_exists_string(
        generic_load_relation_data_delimited(),
        foreign_delimited_list ) )
{
    return;
}

list_set(
    generic_load_relation->foreign_delimited_list,
    generic_load_relation_data_delimited() );

/* Attributes */
char *one_folder_name;
LIST *relation_foreign_key_list;
char *appaserver_table_name;
LIST *foreign_delimited_list;
```

GENERIC_LOAD_ORPHAN_LIST

```
/* Usage */
GENERIC_LOAD_ORPHAN_LIST *
   generic_load_orphan_list_new(
      GENERIC_LOAD_RELATION_LIST *
         generic_load_relation_list );

/* Process */
GENERIC_LOAD_ORPHAN_LIST *
   generic_load_orphan_list_calloc(
      void );

LIST *list = list_new();

for GENERIC_LOAD_RELATION *generic_load_relation in
   generic_load_relation_list->list
{
   list_set(
      list,
      GENERIC_LOAD_ORPHAN *
         generic_load_orphan_new(
            generic_load_relation ) );
}

if ( !list_length( list ) )
   return (GENERIC_LOAD_ORPHAN_LIST *)0;
else
   return this;

/* Usage */
void generic_load_orphan_list_output(
      GENERIC_LOAD_ORPHAN_LIST *
         generic_load_orphan_list );

/* Process */
for GENERIC_LOAD_ORPHAN *generic_load_orphan in
   generic_load_orphan_list->list
{
   void generic_load_orphan_output(
      generic_load_orphan );
}

/* Attribute */
LIST *list;
```

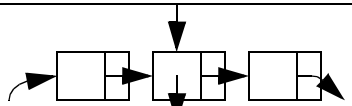


GENERIC_LOAD_ORPHAN

```
/* Usage */
GENERIC_LOAD_ORPHAN *generic_load_orphan_new(
    GENERIC_LOAD_RELATION *generic_load_relation );

/* Process */
GENERIC_LOAD_ORPHAN *
    generic_load_orphan_calloc(
        void );

LIST *foreign_delimited_list = list_new();

for char *foreign_delimited_string in
    generic_load_relation->foreign_delimited_list
{
    LIST *foreign_data_list =
        LIST *list_string_list(
            foreign_delimited_string /* list_string */,
            SQL_DELIMITER );

    char *generic_load_orphan_select_statement(
        generic_load_relation->appaserver_table_name,
        generic_load_relation->
            relation_foreign_key_list,
        foreign_data_list );

    char *generic_load_orphan_input_system(
        char *generic_load_orphan_select_statement() );

    boolean result =
        string_atoi(
            string_pipe_fetch(
                generic_load_orphan_input_system() ) );

    if ( !result )
    {
        list_set(
            foreign_delimited_list,
            foreign_delimited_string );
    }
}

if ( list_length( foreign_delimited_list ) )
    return this;
else
    return (GENERIC_LOAD_ORPHAN *)0;

/* Usage */
char *generic_load_orphan_select_statement(
    char *appaserver_table_name,
    LIST *relation_foreign_key_list,
    LIST *foreign_data_list );

/* Process */
static char select_statement[];
char *ptr = select_statement;
boolean first_time = 1;

ptr +=
    sprintf(
        ptr,
        "select 1 from %s where ",
        appaserver_table_name );

list_rewind( foreign_data_list );

for char *foreign_key in relation_foreign_key_list
{
    if ( first_time )
        first_time = 0;
    else
        ptr += sprintf( ptr, " and " );

    char *foreign_data =
        list_get(
            foreign_data_list );

    ptr +=
        sprintf(
            ptr,
            "%s = '%s'",
            foreign_key,
            foreign_data );

    list_next( foreign_data_list );
}
ptr += sprintf( ptr, ";" );

/* Usage */
void generic_load_orphan_output(
    GENERIC_LOAD_ORPHAN *
        generic_load_orphan );

/* Process */
char *generic_load_orphan_title_string(
    char *generic_load_orphan->
        generic_load_relation->
        one_folder_name );

char *generic_load_orphan_heading_string(
    LIST *generic_load_orphan->
        generic_load_relation->
        relation_foreign_key_list );

char *generic_load_orphan_output_system(
    char SQL_DELIMITER,
    char *generic_load_orphan_title_string(),
    char *generic_load_orphan_heading_string() );

FILE *generic_load_orphan_output_pipe(
    char *generic_load_orphan_output_system() );

for char *foreign_delimited_string in
    generic_load_orphan->
        generic_load_relation->
        foreign_delimited_list
{
    fprintf(    generic_load_orphan_output_pipe(),
                "%s\n",
                foreign_delimited_string );
}

pclose( generic_load_orphan_output_pipe() );

/* Attributes */
GENERIC_LOAD_RELATION *generic_load_relation;
LIST *foreign_delimited_list;
```

GENERIC_LOAD_ROW

```
/* Usage */
LIST *generic_load_row_list(
        char *generic_load_input->load_filename,
        int skip_header_rows_integer,
        DICTIONARY *position_dictionary,
        DICTIONARY *constant_dictionary,
        LIST *folder_attribute_list,
        char *appaserver_table_name() );

/* Process */
LIST *list = list_new();
int row_number = 0;

FILE *generic_load_row_input_file(
    char *load_filename );

for char input[] in string_input(
    generic_load_row_input_file() )
{
    if ( ++row_number <= skip_header_rows_integer )
        continue;

    list_set(
        list,
        GENERIC_LOAD_ROW *
            generic_load_row_new(
                position_dictionary,
                constant_dictionary,
                folder_attribute_list,
                appaserver_table_name,
                input ) );
}
fclose( generic_load_row_input_file() );

/* Usage */
GENERIC_LOAD_ROW *generic_load_row_new(
    DICTIONARY *position_dictionary,
    DICTIONARY *constant_dictionary,
    LIST *folder_attribute_list,
    char *appaserver_table_name,
    char *input );

/* Process */
GENERIC_LOAD_ROW *generic_load_row_calloc( void );

LIST *generic_load_attribute_list(
    position_dictionary,
    constant_dictionary,
    folder_attribute_list,
    input );

if ( !list_length( generic_load_attribute_list ) )
    return (GENERIC_LOAD_ROW *)0;

char *generic_load_attribute_name_list_string(
    generic_load_attribute_list );

char *generic_load_attribute_value_list_string(
    generic_load_attribute_list,
    1 /* include_quotes */ );

char *generic_load_row_insert_statement(
    char *appaserver_table_name,
    char *generic_load_attribute_name_list_string(),
    char *generic_load_attribute_value_list_string() );

free( generic_load_attribute_name_list_string() );
free( generic_load_attribute_value_list_string() );

/* Usage */
void generic_load_row_insert(
    LIST *generic_load_insert->generic_load_row_list,
    char *generic_load_insert_system_string );

/* Process */
FILE *generic_load_row_output_pipe(
    char *generic_load_insert_system_string );

for GENERIC_LOAD_ROW *
    generic_load_row in
        generic_load_row_list
{
    fprintf(
        generic_load_row_output_pipe(),
        "%s\n",
        generic_load_row->insert_statement );
}

pclose( generic_load_row_output_pipe() );

/* Usage */
void generic_load_row_display(
    char *folder_name,
    LIST *generic_load_row_list );

/* Process */
char *generic_load_row_title_string(
    char *folder_name );

GENERIC_LOAD_ROW *generic_load_row_first =
    list_first( generic_load_row_list );

char *generic_load_attribute_heading_string(
    generic_load_row_first->
        generic_load_attribute_list );

char *generic_load_row_output_system(
    char delimiter /* ',' */,
    int HTML_TABLE_QUEUE_TOP_BOTTOM,
    char *generic_load_row_title_string(),
    char *generic_load_attribute_heading_string() );

FILE *generic_load_row_output_pipe(
    generic_load_row_output_system() );

for GENERIC_LOAD_ROW *generic_load_row in
    generic_load_row_list
{
    char *generic_load_attribute_value_list_string(
        generic_load_row->generic_load_attribute_list,
        0 /* not include_quotes */ );

    fprintf(
        generic_load_row_output_pipe(),
        "%s\n",
        generic_load_attribute_value_list_string() );
}

pclose( generic_load_row_output_pipe() );

/* Attributes */
LIST *generic_load_attribute_list;
char *insert_statement;
```



GENERIC_LOAD_ATTRIBUTE

GENERIC_LOAD_ATTRIBUTE

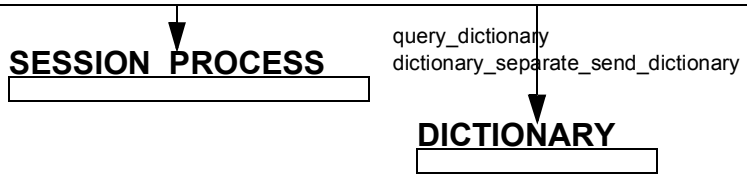
```
/* Usage */
LIST *generic_load_attribute_list(
    DICTIONARY *position_dictionary,
    DICTIONARY *constant_dictionary,
    LIST *folder_attribute_list,
    char *input );

/* Process */
LIST *list = list_new();

for FOLDER_ATTRIBUTE *folder_attribute in
    folder_attribute_list
{
    list_set(
        list,
        GENERIC_LOAD_ATTRIBUTE *
            generic_load_attribute_new(
                position_dictionary,
                constant_dictionary,
                input,
                folder_attribute ) );
}

/* Usage */
GENERIC_LOAD_ATTRIBUTE *
    generic_load_attribute_new(
        DICTIONARY *position_dictionary,
        DICTIONARY *constant_dictionary,
        char *input,
        FOLDER_ATTRIBUTE *folder_attribute );

/* Process */
GENERIC_LOAD_ATTRIBUTE *
    generic_load_attribute_calloc(
        void );

if ( ( char *value =
        dictionary_get(
            folder_attribute->attribute_name,
            constant_dictionary ) ) )
                return this;

if ( ! ( char *position_string =
            dictionary_get(
                folder_attribute->attribute_name,
                position_dictionary ) ) )
                    return (GENERIC_LOAD_ATTRIBUTE *)0;

int position_integer = atoi( position_string );

if ( position_integer < 1 )
    return (GENERIC_LOAD_ATTRIBUTE *)0;

char string_delimiter( input );

if ( !piece(
        char piece_buffer[],
        string_delimiter,
        input,
        position_integer - 1 ) )
{
    return (GENERIC_LOAD_ATTRIBUTE *)0;
}

char *value = strdup( piece_buffer );

/* Usage */
char *generic_load_attribute_name_list_string(
    LIST *generic_load_attribute_list );

/* Process */
char list_string[];
char *ptr = list_string;

for GENERIC_LOAD_ATTRIBUTE *
    generic_load_attribute in
        generic_load_attribute_list
{
    if ( ptr != list_string ) ptr += sprintf( ptr, "," );

    ptr +=
        sprintf(
            ptr,
            "%s",
            generic_load_attribute->
                folder_attribute->
                attribute_name );
}

/* Usage */
char *generic_load_attribute_value_list_string(
    LIST *generic_load_attribute_list,
    boolean include_quotes );

/* Process */
char list_string[];
char *ptr = list_string;

for GENERIC_LOAD_ATTRIBUTE *
        generic_load_attribute in
            generic_load_attribute_list
{
    if ( ptr != list_string ) ptr += sprintf( ptr, "," );

    if (    !include_quotes
    ||      attribute_is_number(
                generic_load_attribute->
                    folder_attribute->
                    attribute->
                    datatype_name ) )
    {
        ptr +=
            sprintf(
                ptr,
                "%s",
                value );
    }
    else
    {
        ptr +=
            sprintf(
                ptr,
                "'%s'",
                value );
    }
}

/* Public */
GENERIC_LOAD_ATTRIBUTE *
    generic_load_attribute_seek(
        LIST *generic_load_attribute_list(),
        char *attribute_name );

char *generic_load_attribute_heading_string(
    LIST *generic_load_attribute_list );

/* Attributes */
FOLDER_ATTRIBUTE *folder_attribute;
char *value;
```

GENERIC_LOAD_INPUT

```
/* Usage */
GENERIC_LOAD_INPUT *generic_load_input_new(
    char *GENERIC_LOAD_FILENAME_LABEL,
    char *GENERIC_LOAD_SKIP_HEADER_ROWS_LABEL,
    char *GENERIC_LOAD_EXECUTE_YN_LABEL,
    char *GENERIC_LOAD_POSITION_PREFIX,
    char *GENERIC_LOAD_CONSTANT_PREFIX,
    DICTIONARY *non_prefixed_dictionary );

/* Process */
GENERIC_LOAD_INPUT *generic_load_input_calloc(
    void );

char *load_filename =
    dictionary_get(
        GENERIC_LOAD_FILENAME_LABEL,
        non_prefixed_dictionary );

if ( !load_filename )
{
    return (GENERIC_LOAD_INPUT *)0;
}

int skip_header_rows_integer =
    string_atoi(
        dictionary_get(
            GENERIC_LOAD_SKIP_HEADER_ROWS_LABEL,
            non_prefixed_dictionary ) );

boolean execute =
    ( *dictionary_get(
            GENERIC_LOAD_EXECUTE_YN_LABEL,
            non_prefixed_dictionary ) == 'y' );

DICTIONARY *position_dictionary =
    dictionary_prefix_extract(
        GENERIC_LOAD_POSITION_PREFIX,
        non_prefixed_dictionary /* source_dictionary */ );

DICTIONARY *constant_dictionary =
    dictionary_prefix_extract(
        GENERIC_LOAD_CONSTANT_PREFIX,
        non_prefixed_dictionary /* source_dictionary */ );

/* Attributes */
char *load_filename;
int skip_header_rows_integer;
boolean execute;
DICTIONARY *position_dictionary;
DICTIONARY *constant_dictionary;
```

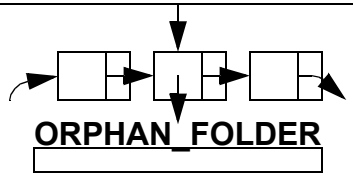
CHANGE_PASSWORD

```
/* Usage */
CHANGE_PASSWORD *change_password_new(
    int argc,
    char **argv,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *process_name );

/* Process */
CHANGE_PASSWORD *change_password_calloc( void );

/* Sets appaserver environment and outputs argv. */
/* Each parameter is security inspected. */
/* Any error will exit( 1 ). */
SESSION_PROCESS *session_process =
    session_process_integrity_exit(
        argc,
        argv,
        application_name,
        session_key,
        login_name,
        role_name,
        process_name );

DICTIONARY *change_password_query_dictionary(
    APPASERVER_USER_LOGIN_NAME,
    QUERY_RELATION_OPERATOR_PREFIX,
    QUERY_EQUAL,
    login_name );

DICTIONARY *dictionary_separate_send_dictionary(
    (DICTIONARY *)0 /* sort_dictionary */,
    DICTIONARY_SEPARATE_SORT_PREFIX,
    change_password_query_dictionary(),
    DICTIONARY_SEPARATE_QUERY_PREFIX,
    (DICTIONARY *)0 /* drillthru_dictionary */,
    DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
    (DICTIONARY *)0 /* ignore_dictionary */,
    DICTIONARY_SEPARATE_IGNORE_PREFIX,
    (DICTIONARY *)0 /* no_display_dictionary */,
    DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
    (DICTIONARY *)0 /* pair_dictionary */,
    DICTIONARY_SEPARATE_PAIR_PREFIX,
    (DICTIONARY *)0 /* non_prefixed_dictionary */ );

char *dictionary_separate_send_string(
    dictionary_separate_send_dictionary() );

char *execute_system_string_table_edit(
    TABLE_EDIT_EXECUTABLE,
    session_key,
    login_name,
    role_name,
    APPASERVER_USER_TABLE,
    FRAMESET_PROMPT_FRAME /* target_frame */,
    (char *)0 /* sub_sub_title */,
    dictionary_separate_send_string(),
    appaserver_error_filename(
        application_name ) );

/* Usage */
DICTIONARY *change_password_query_dictionary(
    const char *APPASERVER_USER_LOGIN_NAME,
    const char *QUERY_RELATION_OPERATOR_PREFIX,
    const char *QUERY_EQUAL,
    char *login_name );

/* Process */
DICTIONARY *query_dictionary = dictionary_small();

void dictionary_set(
    query_dictionary,
    APPASERVER_USER_LOGIN_NAME,
    login_name );

char *query_relation_key(
    QUERY_RELATION_OPERATOR_PREFIX,
    APPASERVER_USER_LOGIN_NAME );

dictionary_set(
    query_dictionary,
    strdup( query_relation_key() ),
    QUERY_EQUAL );

/* Attributes */
SESSION_PROCESS *session_process;
DICTIONARY *query_dictionary;
DICTIONARY *dictionary_separate_send_dictionary;
char *dictionary_separate_send_string;
char *execute_system_string_table_edit;

/* Driver */
system(
    change_password->
        execute_system_string_table_edit );
```



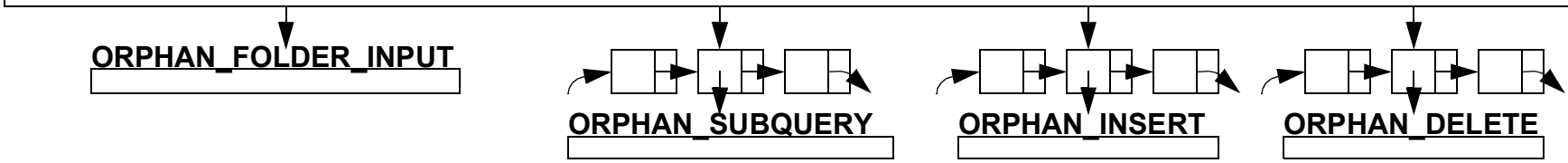
ORPHAN

```
/* Usage */
ORPHAN *orphan_new(
    char *application_name,
    boolean delete_boolean );

/* Process */
ORPHAN *orphan_calloc(
    void );

LIST *orphan_folder_list(
    application_name,
    delete_boolean );

/* Driver */
boolean orphan_folder_clean_boolean(
    orphan->orphan_folder_list );

if ( orphan_folder_clean_boolean() )
{
    void orphan_clean_output(
        boolean stdout_boolean );
}
else
for ORPHAN_FOLDER *orphan_folder in
    orphan->orphan_folder_list
{
    if ( execute_boolean && !delete_boolean )
    {
        orphan_insert_execute(
            orphan_folder->orphan_insert_list );
    }
    else
    if ( execute_boolean && delete_boolean )
    {
        orphan_delete_execute(
            orphan_folder->orphan_delete_list );
    }
    else
    if ( !delete_boolean )
    {
        void orphan_insert_list_display(
            stdout_boolean,
            orphan_folder->folder_name,
            orphan_folder->orphan_insert_list );
    }
    else
    {
        void orphan_delete_list_display(
            stdout_boolean,
            orphan_folder->orphan_delete_list );
    }
}

/* Attributes */
LIST *orphan_folder_list;
```



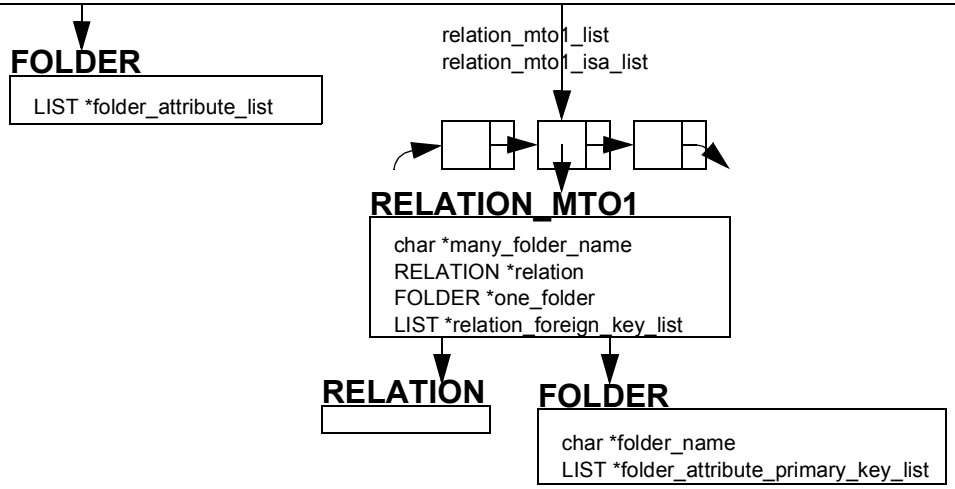
ORPHAN_FOLDER

```
/* Usage */
LIST *orphan_folder_list(
    char *application_name,
    boolean delete_boolean );

/* Process */
LIST *list = list_new();

LIST *folder_fetch_name_list(
    FOLDER_PRIMARY_KEY,
    FOLDER_TABLE );

for char *folder_name in
    folder_fetch_name_list
{
    ORPHAN_FOLDER *orphan_folder =
        orphan_folder_new(
            application_name,
            delete_boolean,
            folder_name );

    list_set(
        list,
        orphan_folder_new() );
}

/* Usage */
ORPHAN_FOLDER *orphan_folder_new(
    char *application_name,
    boolean delete_boolean,
    char *folder_name );

/* Process */
ORPHAN_FOLDER *orphan_folder_calloc(
    void );

ORPHAN_FOLDER_INPUT *orphan_folder_input =
    orphan_folder_input_new(
        folder_name );

LIST *orphan_subquery_list = list_new();

for RELATION_MTO1 *relation_mto1 in
    orphan_folder_input->relation_mto1_list
{
    list_set(
        orphan_subquery_list,
        orphan_subquery_new(
            application_name,
            folder_name,
            relation_mto1 ) );
}

for RELATION_MTO1 *relation_mto1 in
    orphan_folder_input->relation_mto1_isa_list
{
    list_set(
        orphan_subquery_list,
        orphan_subquery_new(
            application_name,
            folder_name,
            orphan_data_list_boolean,
            relation_mto1 ) );
}

if ( !delete_boolean )
{
    LIST *orphan_insert_list(
        orphan_subquery_list );
}
else
{
    LIST *orphan_delete_list(
        orphan_subquery_list );
}

/* Usage */
boolean orphan_folder_clean_boolean(
    LIST *orphan_folder_list );

/* Process */
for ORPHAN_FOLDER *orphan_folder in
    orphan_folder_list
{
    boolean orphan_subquery_clean_boolean(
        orphan_folder->orphan_subquery_list );

    if ( !orphan_subquery_clean_boolean() )
    {
        return 0;
    }
}
return 1;

/* Attributes */
char *folder_name;
ORPHAN_FOLDER_INPUT *orphan_folder_input;
LIST *orphan_subquery_list;
LIST *orphan_insert_list;
LIST *orphan_delete_list;
```



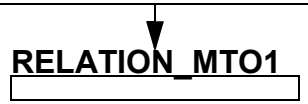
ORPHAN_FOLDER_INPUT

```
/* Usage */
ORPHAN_FOLDER_INPUT *orphan_folder_input_new(
    char *folder_name );

/* Process */
ORPHAN_FOLDER_INPUT *orphan_folder_input_calloc(
    void );

FOLDER *folder_fetch(
    folder_name,
    (LIST *)0 /* exclude_attribute_name_list */,
    1 /* fetch_folder_attribute_list */,
    0 /* not fetch_attribute */,
    1 /* cache_boolean */ );

LIST *relation_mto1_list(
    folder_name
        /* many_folder_name */,
    folder_new()->folder_attribute_primary_key_list
        /* many_folder_primary_key_list */ );

LIST *relation_mto1_isa_list(
    (LIST *)0 /* mto1_isa_list Pass in null */,
    folder_name,
    folder_new()->folder_attribute_primary_key_list,
    0 /* not fetch_relation_one2m_list */,
    0 /* not fetch_relation_mto1_list */ );

/* Attributes */
FOLDER *folder;
LIST *relation_mto1_list;
LIST *relation_mto1_isa_list;
```



ORPHAN_SUBQUERY (1)

```
/* Usage */
ORPHAN_SUBQUERY *orphan_subquery_new(
    char *application_name,
    char *folder_name,
    RELATION_MTO1 *relation_mto1 );

/* Process */
ORPHAN_SUBQUERY *orphan_subquery_calloc(
    void );

char *many_table_name =
    char *strdup(
        appaserver_table_name(
            folder_name ) );

char *one_table_name =
    char *strdup(
        appaserver_table_name(
            relation_mto1->one_folder_name ) );

char *orphan_subquery_string(
    relation_mto1,
    many_table_name,
    one_table_name );

char *orphan_subquery_system_string(
    orphan_subquery_string() );

LIST *orphan_data_list =
    list_pipe(
        orphan_subquery_system_string() );

/* Usage */
char *orphan_subquery_string(
    RELATION_MTO1 *relation_mto1,
    char *many_table_name,
    char *one_table_name );

/* Process */
char string[];
char *ptr = string;
boolean first_time = 1;

ptr +=
    sprintf(
        ptr,
        "select distinct %s from %s ",
        list_display_delimited(
            relation_mto1->relation_foreign_key_list,
            ',' ),
        many_table_name );

ptr +=
    sprintf(
        ptr,
        "where not exists ( select 1 from %s where ",
        one_table_name );

list_rewind(
    relation_mto1->
        one_folder->
        folder_attribute_primary_key_list );

for char *foreign_key in
    relation_mto1->relation_foreign_key_list
{
    char *primary_key =
        list_get(
            relation_mto1->
                one_folder->
                folder_attribute_primary_key_list );

    if ( first_time )
        first_time = 0;
    else
        ptr += sprintf( ptr, " and " );

    ptr += sprintf( ptr,
            "%s.%s = %s.%s and ",
            one_table_name,
            primary_key,
            many_table_name,
            foreign_key );

    list_next(
        relation_mto1->
            one_folder->
            primary_key_list );
}

ptr += sprintf( ptr, ")" );

/*
Joining with nulls in the subquery is producing
false positives.
*/
for char *foreign_key in
    relation_mto1->relation_foreign_key_list
{
    ptr += sprintf( ptr,
            " and %s.%s is not null",
            many_table_name,
            foreign_key );
}
ptr += sprintf( ptr, ")" );

/* Usage */
char *orphan_subquery_system_string(
    char *orphan_subquery_string();

/* Process */
sprintf(
    system_string[],
    "echo \"%s\" | "
    "sql.e | "
    /* subquery returns empty lines for null */
    "grep -v '^$' | "
    /* empty lines for multi-select have leading ^ */
    "grep -v '^\\^'",
    orphan_subquery_string );

/* Attributes */
RELATION_MTO1 *relation_mto1;
char *many_table_name;
char *one_table_name;
char *string;
char *system_string;
LIST *orphan_data_list;
```

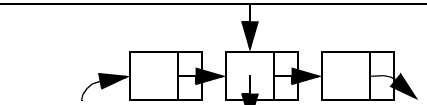


ORPHAN_SUBQUERY (2)

```
/* Usage */
boolean orphan_subquery_clean_boolean(
    LIST *orphan_subquery_list );

/* Process */
for ORPHAN_SUBQUERY *orphan_subquery in
    orphan_subquery_list )
{
    int list_length(
        orphan_subquery->
            orphan_data_list );

    if ( list_length() ) return 0;
}
return 1;

/* Driver */
void orphan_subquery_string_stdout(
    LIST *orphan_subquery_list );

/* Process */
```

ORPHAN_DELETE

```
/* Usage */
LIST *orphan_delete_list(
    LIST *orphan_subquery_list );

/* Process */
LIST *list = list_new();

for ORPHAN_SUBQUERY *orphan_subquery in
    orphan_subquery_list
{
    if ( list_length(
            orphan_subquery->
                orphan_data_list ) )
    {
        list_set(
            list,
            orphan_delete_new(
                orphan_subquery->many_table_name,
                orphan_subquery->one_table_name,
                orphan_subquery->
                    relation_mto1->
                    relation_foreign_key_list,
                orphan_subquery->orphan_data_list ) );
    }
}

/* Usage */
ORPHAN_DELETE *orphan_delete_new(
    char *many_table_name,
    char *one_table_name,
    LIST *relation_foreign_key_list,
    LIST *orphan_data_list );

/* Process */
ORPHAN_DELETE *orphan_delete_calloc(
    void );

LIST *orphan_delete_statement_list(
    many_table_name,
    relation_foreign_key_list,
    orphan_data_list );

/* Usage */
void orphan_delete_list_display(
    boolean stdout_boolean,
    LIST *orphan_delete_list );

/* Process */
for ORPHAN_DELETE *orphan_delete in
    orphan_delete_list
{
    if ( !list_rewind(
            orphan_delete->
                orphan_delete_statement_list ) )
    {
        continue;
    }

    FILE *orphan_delete_display_pipe(
        boolean stdout_boolean,
        char *orphan_delete->one_table_name );

    for ORPHAN_DELETE_STATEMENT *
            orphan_delete_statement in
                orphan_delete->
                    orphan_delete_statement_list
    {
        fprintf(
            orphan_delete_display_pipe(),
            "%s|%s|%s\n",
            orphan_delete_statement->many_table_name,
            orphan_delete_statement->orphan_data,
            orphan_delete_statement->
                delete_key_list_data_string_sql );
    }
    pclose( orphan_delete_display_pipe() );
}

/* Usage */
void orphan_delete_execute(
    LIST *orphan_folder->orphan_delete_list );

/* Process */
char *delete_execute_system_string(
    SQL_EXECUTABLE );

FILE *appaserver_output_pipe(
    delete_execute_system_string() );

for ORPHAN_DELETE *orphan_delete in
    orphan_delete_list
{
    for ORPHAN_DELETE_STATEMENT *
        orphan_delete_statement in
            orphan_delete->
                orphan_delete_statement_list
    {
        fprintf(
            appaserver_output_pipe(),
            "%s\n",
            orphan_delete_statement->
                delete_key_list_data_string_sql );
    }
}
pclose( appaserver_output_pipe() );

/* Attribute */
char *many_table_name;
char *one_table_name;
LIST *relation_foreign_key_list;
LIST *orphan_delete_statement_list;
```



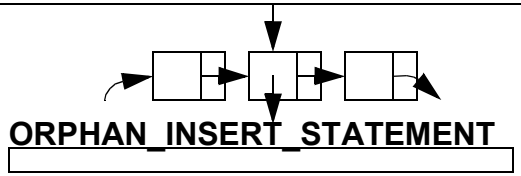
ORPHAN_DELETE_STATEMENT

ORPHAN_DELETE_STATEMENT

```
/* Usage */
LIST *orphan_delete_statement_list(
    char *many_table_name,
    LIST *relation_mto1->relation_foreign_key_list,
    LIST *orphan_subquery->orphan_data_list );

/* Process */
LIST *list = list_new();

for char *orphan_data in orphan_data_list
{
    list_set(
        list,
        orphan_delete_statement_new(
            SQL_DELIMITER,
            many_table_name,
            relation_foreign_key_list,
            orphan_data ) );
}

/* Usage */
ORPHAN_DELETE_STATEMENT *
    orphan_delete_statement_new(
        const char SQL_DELIMITER,
        char *many_table_name,
        LIST *relation_foreign_key_list,
        char *orphan_data );

/* Process */
ORPHAN_DELETE_STATEMENT *
    orphan_delete_statement_calloc(
        void );

char *delete_key_list_data_string_sql(
    SQL_DELIMITER,
    many_table_name,
    relation_foreign_key_list,
    orphan_data );

/* Attributes */
char *many_table_name;
char *orphan_data;
char *delete_key_list_data_string_sql;
```

ORPHAN_INSERT

```
/* Usage */
LIST *orphan_insert_list(
    LIST *orphan_subquery_list );

/* Process */
LIST *list = list_new();

for ORPHAN_SUBQUERY *orphan_subquery in
    orphan_subquery_list
{
    if ( list_length(
            orphan_subquery->
                orphan_data_list ) )
    {
        list_set(
            list,
            orphan_insert_new(
                orphan_subquery ) );
    }
}

/* Usage */
ORPHAN_INSERT *orphan_insert_new(
    ORPHAN_SUBQUERY *orphan_subquery );

/* Process */
ORPHAN_INSERT *orphan_insert_calloc(
    void );

LIST *orphan_insert_statement_list(
    orphan_subquery->one_table_name,
    orphan_subquery->
        relation_mto1->
            one_folder->
            primary_key_list,
    orphan_subquery->orphan_data_list );

* Usage */
void orphan_insert_list_display(
    boolean stdout_boolean,
    char *folder_name,
    LIST *orphan_insert_list );

/* Process */
for ORPHAN_INSERT *orphan_insert in
    orphan_insert_list
{
    if ( !list_rewind(
            orphan_insert->
                orphan_insert_statement_list ) )
    {
        continue;
    }

    FILE *orphan_insert_display_pipe(
        boolean stdout_boolean,
        char *folder_name );

    for ORPHAN_INSERT_STATEMENT *
            orphan_insert_statement in
                orphan_insert->
                    orphan_insert_statement_list
    {
        fprintf(
            orphan_insert_display_pipe(),
            "%s|%s|%s\n",
            orphan_insert_statement->one_table_name,
            orphan_insert_statement->orphan_data,
            orphan_insert_statement->statement );
    }
    pclose( orphan_insert_display_pipe() );
}

/* Usage */
void orphan_insert_execute(
    LIST *orphan_folder->orphan_insert_list );

/* Process */

/* Attribute */
LIST *orphan_insert_statement_list;
```

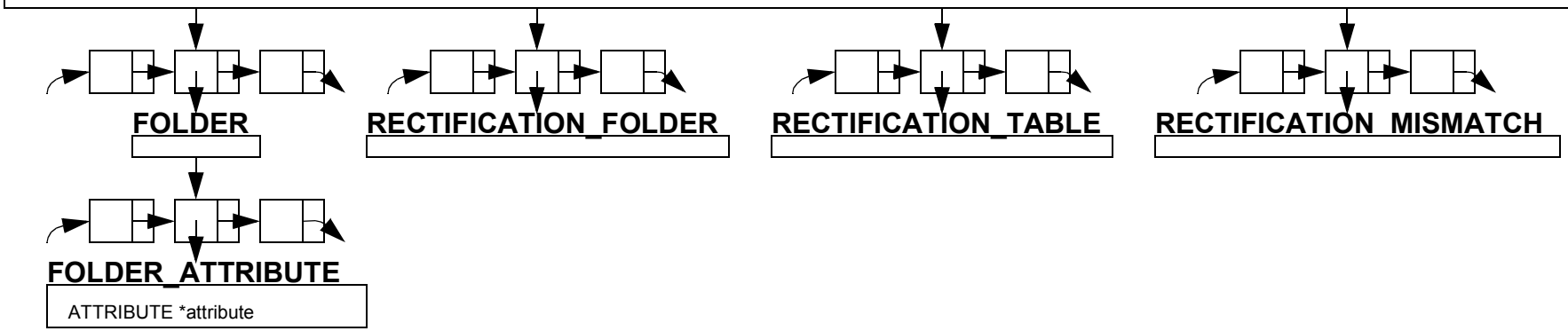


ORPHAN_INSERT_STATEMENT

```
/* Usage */
LIST *orphan_insert_statement_list(
    char *one_table_name,
    LIST *folder_attribute_primary_key_list,
    LIST *orphan_data_list );

/* Process */
LIST *list = list_new();
LIST *existing_data_list = list_new();

for char *orphan_data in orphan_data_list
{
    if ( list_string_exists(
            orphan_data,
            existing_data_list ) )
    {
        continue;
    }

    list_set(
        list,
        orphan_insert_statement_new(
            one_table_name,
            folder_attribute_primary_key_list,
            orphan_data ) );

    list_set(
        existing_data_list,
        orphan_data );
}

/* Usage */
ORPHAN_INSERT_STATEMENT *
    orphan_insert_statement_new(
        char *one_table_name,
        LIST *folder_attribute_primary_key_list,
        char *orphan_data );

/* Process */
char statement[];
char *ptr = statement;

ORPHAN_INSERT_STATEMENT *
    orphan_insert_statement_calloc(
        void );

ptr +=
    sprintf(
        ptr,
        "insert into %s (%s) values ( ",
        one_table_name,
        list_display_delimited(
            folder_attribute_primary_key_list,
            ',' ) );

for (  int p = 0;
        piece(
            char data[],
            SQL_DELIMITER,
            orphan_data,
            p );
        p++ )
{
    if ( p ) ptr += sprintf( ptr, ", " );

    ptr +=
        sprintf(
            ptr,
            "'%s'",
            data );
}

ptr += sprintf( ptr, " );" );

/* Attributes */
char *one_table_name;
char *orphan_data;
char *statement;
```

RECTIFICATION (1)

```
/* Usage */
RECTIFICATION *rectification_fetch(
    char *application_name,
    char *security_sql_injection_escape(
        session_key ),
    char *security_sql_injection_escape(
        login_name ) );

/* Process */
RECTIFICATION *rectification_calloc(
    void );

LIST *folder_list(
    (LIST *)0 /* exclude_attribute_name_list */,
    1 /* fetch_folder_attribute_list */,
    1 /* fetch_attribute */ );

LIST *rectification_folder_list(
    application_name,
    folder_list() );

LIST *rectification_table_list(
    application_name );

LIST *rectification_mismatch_list = list_new();

for RECTIFICATION_FOLDER *rectification_folder in
    rectification_folder_list()
{
    RECTIFICATION_TABLE *rectification_table =
        rectification_table_seek(
            rectification_folder->appaserver_table_name,
            rectification_table_list() );

    if ( !rectification_table )
    {
        list_set(
            rectification_mismatch_list,
            RECTIFICATION_MISMATCH *
                rectification_mismatch_no_table_new(
                    rectification_folder->
                        folder_name ) );

        continue;
    }

    list_set(
        rectification_mismatch_list,
        RECTIFICATION_MISMATCH *
            rectification_mismatch_new(
                application_name,
                session_key,
                login_name,
                ROLE_SYSTEM /* role_name */,
                rectification_folder->folder_name,
                rectification_folder->
                    rectification_attribute_list,
                rectification_table->
                    rectification_column_list ) );
}

for RECTIFICATION_TABLE *rectification_table in
    rectification_table_list()
{
    RECTIFICATION_FOLDER *rectification_folder =
        rectification_folder_seek(
            rectification_table->table_name,
        rectification_folder_list() );

    if ( !rectification_folder )
    {
        list_set(
            rectification_mismatch_list,
            RECTIFICATION_MISMATCH *
                rectification_mismatch_no_folder_new(
                    application_name,
                    session_key,
                    login_name,
                    ROLE_SYSTEM /* role_name */,
                    rectification_table->
                        table_name ) );
    }
}

LIST *rectification_not_in_role_table_name_list(
    FOLDER_TABLE,
    ROLE_FOLDER_TABLE,
    FOLDER_PRIMARY_KEY );

/* Attributes */
LIST *folder_list;
LIST *rectification_folder_list;
LIST *rectification_table_list;
LIST *rectification_mismatch_list;
LIST *not_in_role_table_name_list;
```



RECTIFICATION (2)

```
/* Usage */
LIST *rectification_not_in_role_table_name_list(
    const char *FOLDER_TABLE,
    const char *ROLE_FOLDER_TABLE,
    const char *FOLDER_PRIMARY_KEY );

/* Process */
char *rectification_not_in_role_table_system_string(
    FOLDER_TABLE,
    ROLE_FOLDER_TABLE,
    FOLDER_PRIMARY_KEY );
```

```
LIST *list_pipe(
    rectification_not_in_role_table_system_string() );

/* Usage */
char *rectification_not_in_role_table_system_string(
    const char *FOLDER_TABLE,
    const char *ROLE_FOLDER_TABLE,
    const char *FOLDER_PRIMARY_KEY );

/* Process */
```

```
char *rectification_not_in_role_table_where(
    const char *ROLE_FOLDER_TABLE,
    const char *FOLDER_PRIMARY_KEY );

snprintf(
    static char system_string[],
    sizeof ( system_string ),
    “select.sh %s %s \”%s\””,
    FOLDER_PRIMARY_KEY,
    FOLDER_TABLE,
    rectification_not_in_role_table_where() );
```

RECTIFICATION_MISMATCH (1)

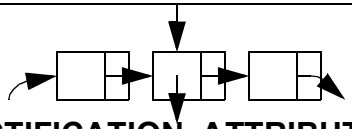
```
/* Usage */
RECTIFICATION_MISMATCH *
   rectification_mismatch_new(
      char *application_name,
      char *session_key,
      char *login_name,
      char *role_name,
      char *folder_name,
      LIST *rectification_attribute_list,
      LIST *rectification_column_list );

/* Process */
RECTIFICATION_MISMATCH *
   rectification_mismatch_calloc(
      void );

for RECTIFICATION_ATTRIBUTE *rectification_attribute
   in rectification_attribute_list
{
   RECTIFICATION_COLUMN *
      rectification_column =
         rectification_column_seek(
            rectification_attribute->
               attribute_name
                  /* column_name */,
            rectification_column_list );

   if ( !rectification_column )
   {
      missing_column_name =
         rectification_attribute->
            attribute_name;

      return this;
   }

   boolean rectification_mismatch_width_boolean(
      int rectification_attribute->width,
      int rectification_column->width );

   boolean rectification_mismatch_float_boolean(
      int rectification_attribute->float_decimal_places,
      int rectification_column->float_decimal_places );

   if (rectification_mismatch_width_boolean()
   ||  rectification_mismatch_float_boolean() )
         return this;

   boolean rectification_mismatch_key_boolean(
      int rectification_attribute->primary_key_index,
      int rectification_column->primary_key_index );

   if ( rectification_mismatch_key_boolean() )
         return this;

   boolean rectification_mismatch_datatype_boolean(
      char *rectification_attribute->datatype_name,
      int rectification_attribute->width,
      char *rectification_column->
         attribute_datatype_name );

   if ( rectification_mismatch_datatype_boolean() )
         return this;
}

for RECTIFICATION_COLUMN *rectification_column in
   rectification_column_list
{
   RECTIFICATION_ATTRIBUTE *
      rectification_attribute =
         rectification_attribute_seek(
            rectification_column->
               column_name
                  /* attribute_name */,
            rectification_attribute_list );

   if ( !rectification_attribute )
   {
      char *missing_attribute_name =
         rectification_column->
            column_name;

      char *drop_column_href_string(
         DROP_COLUMN_EXECUTABLE,
         application_name,
         session_key,
         login_name,
         role_name,
         folder_name,
         missing_attribute_name );

      char *rectification_mismatch_column_prompt(
         char *missing_attribute_name );

      char *rectification_mismatch_drop_column_tag(
         char *drop_column_href_string(),
         char *target_frame
            /* FRAMESET_TABLE_FRAME */,
         char *
            rectification_mismatch_column_prompt() );

      return this;
   }
}
return NULL;

/* Attributes */
char *folder_name;
RECTIFICATION_ATTRIBUTE *rectification_attribute;
RECTIFICATION_COLUMN *rectification_column;
char *missing_column_name;
boolean width_boolean;
boolean float_boolean;
boolean key_boolean;
boolean datatype_boolean;
char *missing_attribute_name;
char *drop_column_href_string;
char *drop_column_tag;
char *missing_table;
char *missing_folder;
char *drop_table_href_string;
char *drop_table_tag;
```

RECTIFICATION_MISMATCH (2)

```
/* Usage */
RECTIFICATION_MISMATCH *
    rectification_mismatch_no_table_new(
        char *rectification_folder->folder_name );

/* Process */
RECTIFICATION_MISMATCH *
    rectification_mismatch_calloc();

char *missing_table = folder_name;

/* Usage */
RECTIFICATION_MISMATCH *
    rectification_mismatch_no_folder_new(
        char *application_name,
        char *session_key,
        char *login_name,
        char *role_name,
        char *rectification_table->table_name );

/* Process */
RECTIFICATION_MISMATCH *
    rectification_mismatch_calloc();

char *missing_folder = table_name;

char *drop_table_href_string(
    DROP_TABLE_EXECUTABLE,
    application_name,
    session_key,
    login_name,
    role_name,
    missing_folder /* folder_name */ );

char *rectification_mismatch_table_prompt(
    char *missing_folder );

char *rectification_mismatch_drop_table_tag(
    char *drop_table_href_string(),
    char *target_frame
        /* FRAMESET_TABLE_FRAME */,
    char *rectification_mismatch_table_prompt() );
```

RECTIFICATION_FOLDER

```
/* Usage */
LIST *rectification_folder_list(
    LIST *folder_list() );

/* Process */
LIST *list = list_new();

for FOLDER *folder in folder_list
{
    if ( list_length( folder->folder_attribute_list ) )
    {
        list_set(
            list,
            RECTIFICATION_FOLDER *
                rectification_folder_new(
                    folder->folder_name,
                    folder->folder_attribute_list ) );
    }
}

/* Usage */
RECTIFICATION_FOLDER *rectification_folder_new(
    char *folder_name,
    LIST *folder_attribute_list );

/* Process */
RECTIFICATION_FOLDER *
    rectification_folder_calloc(
        void );

char *strdup(
    appaserver_table_name(
        folder_name ) );

LIST *rectification_attribute_list(
    folder_attribute_list );

/* Usage */
RECTIFICATION_FOLDER *
    rectification_folder_seek(
        char *appaserver_table_name,
        LIST *rectification_folder_list() );

/* Process */

/* Attributes */
char *folder_name;
char *appaserver_table_name;
LIST *rectification_attribute_list;
```

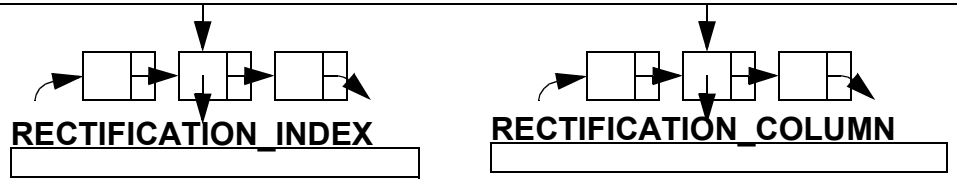


RECTIFICATION_ATTRIBUTE

```
/* Usage */
LIST *rectification_attribute_list(
    LIST *folder_attribute_list );

/* Process */
LIST *list = list_new();

for FOLDER_ATTRIBUTE *folder_attribute in
    folder_attribute_list
{
    list_set(
        list,
        RECTIFICATION_ATTRIBUTE *
            rectification_attribute_new(
                folder_attribute->
                    attribute_name,
                folder_attribute->
                    primary_key_index,
                folder_attribute->
                    attribute->
                    datatype_name,
                folder_attribute->
                    attribute->
                    width,
                folder_attribute->
                    attribute->
                    float_decimal_places ) );
}

/* Usage */
RECTIFICATION_ATTRIBUTE *
    rectification_attribute_new(
        char *attribute_name,
        int primary_key_index,
        char *datatype_name,
        int width,
        int float_decimal_places );

/* Process */
RECTIFICATION_ATTRIBUTE *
    rectification_attribute_calloc(
        void );

/* Usage */
RECTIFICATION_ATTRIBUTE *
    rectification_attribute_seek(
        char *attribute_name,
        LIST *rectification_attribute_list() );

/* Attributes */
char *attribute_name;
int primary_key_index;
char *datatype_name;
int width;
int float_decimal_places;
```

RECTIFICATION_TABLE

```
/* Usage */
LIST *rectification_table_list(
    void );

/* Process */
LIST *list = list_new();

LIST *application_table_name_list();

for char *table_name in
    application_table_name_list() )
{
    list_set(
        list,
        RECTIFICATION_TABLE *
            rectification_table_new(
                table_name ) );
}

/* Usage */
RECTIFICATION_TABLE *rectification_table_new(
    char *table_name );

/* Process */
RECTIFICATION_TABLE *rectification_table_calloc(
    void );

LIST *rectification_index_list(
    CREATE_TABLE_UNIQUE_SUFFIX,
    table_name );

LIST *rectification_column_list(
    rectification_index_list(),
    table_name );

/* Usage */
RECTIFICATION_TABLE *rectification_table_seek(
    char *table_name,
    LIST *rectification_table_list() );

/* Process */

/* Attributes */
char *table_name;
LIST *rectification_index_list;
LIST *rectification_column_list;
```



RECTIFICATION_INDEX

```
/* Usage */
LIST *rectification_index_list(
    const char *CREATE_TABLE_UNIQUE_SUFFIX,
    char *table_name );

/* Process */
LIST *list = list_new();

char *rectification_index_system_string(
    table_name );

FILE *appaserver_input_pipe(
    rectification_index_system_string() );

for string_input( input[], appaserver_input_pipe() )
{
    list_set(
        list,
        RECTIFICATION_INDEX *rectification_index_parse(
            CREATE_TABLE_UNIQUE_SUFFIX,
            table_name,
            input ) );
}
pclose( appaserver_input_pipe() );

/* Usage */
RECTIFICATION_INDEX *rectification_index_parse(
    const char *CREATE_TABLE_UNIQUE_SUFFIX,
    char *table_name,
    char *input );

/* Process */
piece( index_name[], SQL_DELIMITER, input, 2 );

char *create_table_unique_index_name(
    CREATE_TABLE_UNIQUE_SUFFIX,
    table_name );

int strcmp(
    index_name,
    create_table_unique_index_name() );

if ( strcmp() == 0 ) goto rectification_index_continue;

char *create_table_unique_index_name(
    (char *)0 /* CREATE_TABLE_UNIQUE_SUFFIX */,
    table_name );

int strcmp(
    index_name,
    create_table_unique_index_name() );

if ( strcmp() != 0 ) return NULL;

rectification_index_continue:

piece( column_name[], SQL_DELIMITER, input, 4 );

RECTIFICATION_INDEX *rectification_index_new(
    strdup( column_name ) );

piece( primary_string[], SQL_DELIMITER, input, 3 );

int rectification_index_primary_key_index(
    char *primary_string );

/* Usage */
RECTIFICATION_INDEX *rectification_index_new(
    char *column_name );

/* Process */
RECTIFICATION_INDEX *rectification_index_calloc(
    void );

/* Usage */
char *rectification_index_system_string(
    char *table_name );

/* Process */
sprintf( static system_string[],
    "echo \"show index from %s;\" | sql.e",
    table_name );

/* Usage */
RECTIFICATION_INDEX *rectification_index_seek(
    LIST *rectification_index_list(),
    char *column_name );

/* Process */

/* Attributes */
char *column_name;
int primary_key_index;
```

RECTIFICATION_COLUMN

```
/* Usage */
LIST *rectification_column_list(
    LIST *rectification_index_list(),
    char *table_name );

/* Process */
LIST *list = list_new();

char *rectification_column_system_string(
    table_name );

FILE *appaserver_input_pipe(
    rectification_column_system_string() );

for string_input( input[], appaserver_input_pipe() )
{
    list_set(
        list,
        RECTIFICATION_COLUMN *
            rectification_column_parse(
                rectification_index_list,
                input ) );
}
pclose( appaserver_input_pipe() );

/* Usage */
RECTIFICATION_COLUMN *rectification_column_parse(
    LIST *rectification_index_list,
    char *input );

/* Process */
piece( column_name[], SQL_DELIMITER, input, 0 );

RECTIFICATION_COLUMN *rectification_column_new(
    strdup( column_name ) );

piece( datatype_text[], SQL_DELIMITER, input, 1 );
char *strdup( datatype_text );

RECTIFICATION_INDEX *rectification_index_seek(
    rectification_index_list,
    column_name );

if ( rectification_index_seek() )
{
    rectification_column_new()->primary_key_index =
        rectification_index_seek()->
            primary_key_index;
}

int rectification_column_width(
    char *datatype_text );

int rectification_column_float_decimal_places(
    char *datatype_text );

char *rectification_column_datatype_name(
    char *datatype_text );

/* Usage */
RECTIFICATION_COLUMN *rectification_column_new(
    char *column_name );

/* Process */
RECTIFICATION_COLUMN *rectification_column_calloc(
    void );

/* Usage */
char *rectification_column_system_string(
    char *table_name );

/* Process */
sprintf( static system_string[],
    "echo \"describe %s;\" | sql.e",
    table_name );

/* Usage */
RECTIFICATION_COLUMN *rectification_column_seek(
    char *column_name,
    LIST *rectification_column_list() );

/* Process */

/* Attributes */
char *column_name;
char *datatype_text;
int primary_key_index;
int width;
int float_decimal_places;
char *datatype_name;
```

HISTOGRAM

```
/* Usage */
HISTOGRAM *histogram_new(
    LIST *query_fetch_row_list,
    char *number_attribute_name,
    char *title,
    char *where_string,
    char *histogram_output_pdf );

/* Process */
HISTOGRAM *histogram_calloc( void );

LIST *histogram_number_list(
    query_fetch_row_list,
    number_attribute_name );

char *query_where_display(
    where_string,
    80 /* max_length */ );

char *histogram_system_string(
    title,
    histogram_output_pdf,
    query_where_display() );

/* Usage */
LIST *histogram_number_list(
    LIST *query_fetch_row_list,
    char *number_attribute_name );

/* Process */
LIST *number_list = list_new();

for QUERY_ROW *query_row in query_fetch_row_list
{
    QUERY_CELL *query_cell =
        query_cell_seek(
            number_attribute_name,
            query_row->cell_list );

    list_set(
        number_list,
        query_cell->select_datum );
}

/* Usage */
char *histogram_system_string(
    char *title,
    char *histogram_output_pdf,
    char *query_where_display() );

/* Process */
sprintf(
    system_string[],
    "r_histogram.sh %s \"%s\" \"%s\"",
    histogram_output_pdf,
    title,
    query_where_display() /* sub_title */ );

/* Driver */
void histogram_pdf_create(
    LIST *histogram->number_list,
    char *histogram->system_string );

/* Process */
FILE *appaserver_output_pipe(
    system_string );

for char *number in number_list
{
    fprintf(
        appaserver_output_pipe(),
        "%s\n",
        number );
}

void pclose( appaserver_output_pipe );

/* Attributes */
LIST *number_list;
char *query_where_display;
char *system_string;
```

GRACE

```
/* Usage */
char *grace_ps_system_string(
    char *GRACE_EXECUTABLE_TEMPLATE,
    GRACE_FILENAME *grace_filename );

/* Process */
sprintf(    char grace_executable_system_string[],
            GRACE_EXECUTABLE_TEMPLATE,
            grace_filename->grace_home_directory,
            grace_filename->grace_execution_directory );

sprintf(    char system_string[],
            "tee %s |"
            "%s "
            "-noask "
            "-autoscale x "
            "-hdevice PostScript "
            "-pipe "
            "-hardcopy "
            "-printfile %s)",
            grace_filename->agr_output_filename,
            grace_executable_system_string,
            grace_filename->postscript_filename );

/* Usage */
char *grace_pdf_system_string(
    GRACE_FILENAME *grace_filename );

/* Process */
sprintf(    char system_string[],
            “ps2pdf.sh %s %s”,
            grace_filename->postscript_filename,
            grace_filename->pdf_output_filename );

/* Driver */
void grace_single_graph_pdf_create(
    char *grace_ps_system_string,
    GRACE_PAGE *grace_page,
    GRACE_TITLE *grace_title,
    GRACE_GRAPH *grace_graph,
    char *grace_pdf_system_string );

/* Process */
FILE *appaserver_output_pipe(
    grace_ps_system_string );

void grace_page_output(
    grace_page->width_pixels,
    grace_page->length_pixels,
    appaserver_output_pipe() );

void grace_graph_output(
    grace_title,
    grace_graph,
    appaserver_output_pipe() );

void pclose( appaserver_output_pipe() );

system( grace_pdf_system_string );

/* Driver */
void grace_graph_list_pdf_create(
    char *grace_ps_system_string,
    GRACE_PAGE *grace_page,
    GRACE_TITLE *grace_title,
    LIST *grace_graph_list,
    char *grace_pdf_system_string );

/* Process */
FILE *appaserver_output_pipe(
    grace_ps_system_string );

void grace_page_output(
    grace_page->width_pixels,
    grace_page->length_pixels,
    appaserver_output_pipe() );

for GRACE_GRAPH *grace_graph in grace_graph_list
{
    if ( list_at_last( grace_graph_list ) )
    {
        void grace_graph_output(
            grace_title,
            grace_graph,
            appaserver_output_pipe() );
    }
    else
    {
        void grace_graph_output(
            (GRACE_TITLE *)0,
            grace_graph,
            appaserver_output_pipe() );
    }
}

void pclose( appaserver_output_pipe() );

system( grace_pdf_system_string );
```

GRACE_WINDOW

```
/* Usage */
GRACE_WINDOW *grace_window_new(
    GRACE_FILENAME *grace_filename,
    char *sub_title,
    char *filename_key,
    pid_t process_id );

/* Process */
GRACE_WINDOW *grace_window_calloc(
    void );

if ( grace_filename )
{
    char *grace_prompt_pdf_target(
        filename_key,
        process_id );

    char *onload_string =
        static char *chart_window_onload_string(
            grace_filename->pdf_prompt_filename,
            grace_prompt_pdf_target() );
}

char *document_body_onload_open_tag(
    JAVASCRIPT_WAIT_OVER,
    onload_string /* javascript_replace */,
    (char *)0 /* menu_onload_string */ );

char *date_time_now( date_utc_offset() );

char *grace_window_screen_title(
    char *date_time_now() );

char *chart_window_sub_title_display(
    char *sub_title );

char *grace_window_grace_home_url_html(
    char *GRACE_WINDOW_GRACE_URL );

char *html =
    char *chart_window_html(
        document_body_onload_open_tag(),
        grace_window_screen_title(),
        grace_window_sub_title_display() );

free( document_body_onload_open_tag() );

free( chart_window_sub_title_display() );

/* Usage */
char *grace_window_html(
    char *document_body_onload_open_tag,
    char *screen_title,
    char *sub_title_display );

/* Process */
sprintf( html[],
    "%s\n%s\n%s",
    document_body_onload_open_tag,
    screen_title,
    sub_title_display );

/* Driver */
printf(   "%s\n", grace_window->html );

printf( "%s\n", grace_prompt->html );

printf( "%s\n", grace_window->grace_home_url_html );

/* Attributes */
char *onload_string;
char *date_time_now;
char *screen_title;
char *grace_home_url_html;
char *html;
```

GRACE_PROMPT

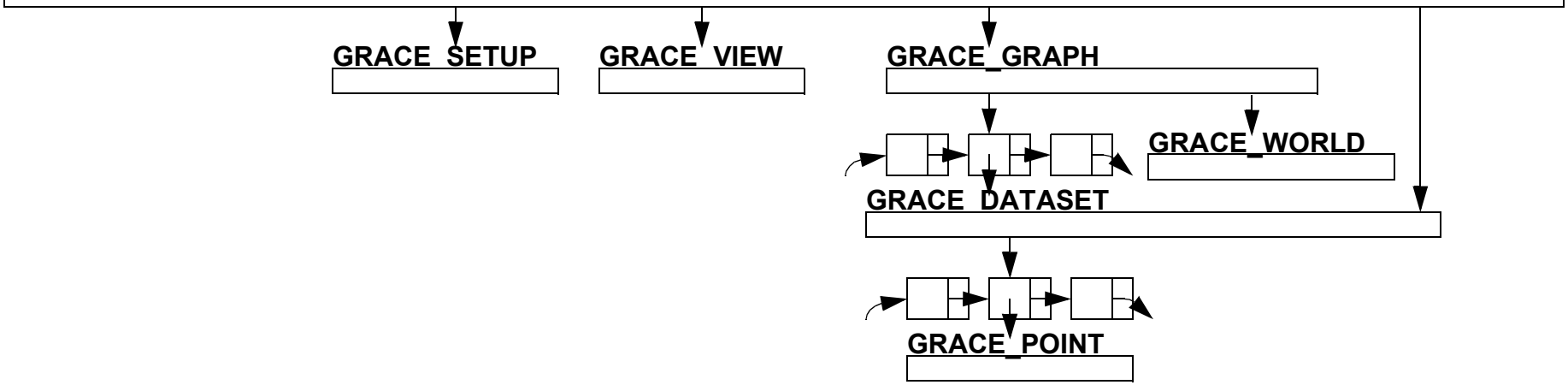
```
/* Usage */
GRACE_PROMPT *grace_prompt_new(
    GRACE_FILENAME *grace_filename,
    char *filename_key,
    pid_t process_id );

/* Process */
GRACE_PROMPT *grace_prompt_calloc(
    void );

char *grace_prompt_pdf_target(
    filename_key,
    process_id );

char *grace_prompt_pdf_anchor_html(
    char *grace_filename->pdf_prompt_filename,
    char *filename_key,
    char *grace_prompt_pdf_target() );

char *grace_prompt_agr_target(
    filename_key,
    process_id );

char *grace_prompt_agr_anchor_html(
    char *grace_filename->agr_prompt_filename,
    char *filename_key,
    char *grace_prompt_agr_target() );

char *grace_prompt_html(
    grace_prompt_pdf_anchor_html(),
    grace_prompt_agr_anchor_html() );

/* Usage */
char *grace_prompt_pdf_target(
    char *filename_key,
    pid_t process_id );

/* Process */
if ( filename_key )
{
    sprintf( static target[],
        "grace_pdf_%s_%d",
        filename_key,
        process_id );
}
else
{
    sprintf( static target[],
        "grace_pdf_%d",
        process_id );
}

/* Usage */
char *grace_prompt_agr_target(
    char *filename_key,
    pid_t process_id );

/* Process */
if ( filename_key )
{
    sprintf( static target[],
        "grace_agr_%s_%d",
        filename_key,
        process_id );
}
else
{
    sprintf( static target[],
        "grace_agr_%d",
        process_id );
}

/* Usage */
char *grace_prompt_html(
    char *grace_prompt_pdf_anchor_html(),
    char *grace_prompt_agr_anchor_html() );

/* Process */
sprintf( html[],
    "<tr><td>%s<td>%s",
    grace_prompt_pdf_anchor_html,
    grace_prompt_agr_anchor_html );

/* Attributes */
char *pdf_target;
char *pdf_anchor_html;
char *agr_target;
char *agr_anchor_html;
char *html;
```

GRACE_HISTOGRAM

```
/* Usage */
GRACE_HISTOGRAM *grace_histogram_new(
    char *application_name,
    char *session_key,
    char *data_directory,
    char *title,
    char *sub_title,
    HISTOGRAM *histogram );

/* Process */
GRACE_HISTOGRAM *grace_histogram_calloc( void );

GRACE_SETUP *grace_setup =
    grace_setup_new(
        application_name,
        session_key,
        data_directory,
        title,
        sub_title,
        (char *)0 /* datatype_name */,
        1 /* grace_graph_list_length */ );

#define NOT_DEFINED
GRACE_VIEW *grace_view =
    grace_view_new(
        GRACE_VIEW_MIN_X,
        GRACE_VIEW_PORTRAIT_X_OFFSET,
        GRACE_VIEW_LANDSCAPE_X_OFFSET,
        1 /* grace_graph_list_length */,
        histogram->
            histogram_segment_range->
            width /* horizontal_range_integer */,
        histogram->
            histogram_segment_range->
            height /* vertical_range */,
        grace_setup->grace_page->landscape_boolean );
#endif

GRACE_VIEW *grace_view =
    grace_view_new(
        GRACE_VIEW_MIN_X,
        GRACE_VIEW_PORTRAIT_X_OFFSET,
        GRACE_VIEW_LANDSCAPE_X_OFFSET,
        1 /* grace_graph_list_length */,
        0 /* horizontal_range_integer */,
        histogram->
            histogram_segment_range->
            height /* vertical_range */,
        grace_setup->grace_page->landscape_boolean );

GRACE_GRAPH *grace_graph =
    grace_histogram_graph_new(
        histogram->number_attribute_name,
        histogram->histogram_segment_range,
        histogram->histogram_segment_list,
        grace_view );

/* Attributes */
GRACE_SETUP *grace_setup;
GRACE_VIEW *grace_view;
GRACE_GRAPH *grace_graph;
```

GRACE_UNITLESS (1)

```
/* Usage */
GRACE_UNITLESS *grace_unitless_new(
    char *application_name,
    char *session_key,
    char *data_directory,
    char *sub_title,
    char *date_attribute_name,
    char *time_attribute_name,
    LIST *query_fetch_row_list,
    int horizontal_range_integer,
    char *title,
    char *grace_datatype_name,
    char *chart_filename_key() );

/* Process */
GRACE_UNITLESS *grace_unitless_calloc( void );

GRACE_SETUP *grace_setup =
    grace_setup_new(
        application_name,
        session_key,
        data_directory,
        title,
        sub_title,
        grace_datatype_name,
        chart_filename_key,
        1 /* grace_graph_list_length */ );

LIST *grace_dataset_point_list(
    date_attribute_name,
    time_attribute_name,
    query_fetch_row_list,
    grace_datatype_name );

GRACE_DATASET *grace_dataset =
    grace_dataset_new(
        (char *)0 /* datatype_name */,
        grace_dataset_point_list(),
        grace_dataset_bar,
        grace_dataset_color_number(
            GRACE_DATASET_MAX_COLOR_NUMBER,
            0 /* not cycle_colors_boolean */ ),
        GRACE_DATASET_LINE_SYMBOL_SIZE ) );

GRACE_VIEW *grace_view =
    grace_view_new(
        GRACE_VIEW_MIN_X,
        GRACE_VIEW_PORTRAIT_X_OFFSET,
        GRACE_VIEW_LANDSCAPE_X_OFFSET,
        1 /* grace_graph_list_length */,
        horizontal_range_integer,
        grace_setup->
            grace_page->
            landscape_boolean );

GRACE_GRAPH *grace_graph =
    grace_unitless_grace_graph(
        horizontal_range_integer,
        grace_view,
        grace_dataset );

GRACE_WORLD *grace_graph->grace_world =
    grace_world_new(
        grace_graph->grace_dataset_list );

/* Attributes */
GRACE_SETUP *grace_setup;
GRACE_DATASET *grace_dataset;
GRACE_VIEW *grace_view;
GRACE_GRAPH *grace_graph;
```



GRACE_UNITLESS (2)

```
/* Usage */
GRACE_GRAPH *grace_unitless_grace_graph(
    int horizontal_range,
    GRACE_VIEW *grace_view,
    GRACE_DATASET *grace_dataset );

/* Process */
```

```
GRACE_GRAPH *grace_graph =
        grace_graph_new(
            0 /* graph_number */,
            1 /* xaxis_tick_on_boolean */,
            horizontal_range,
            (char *)0 /* yaxis_heading */,
            grace_view );
```

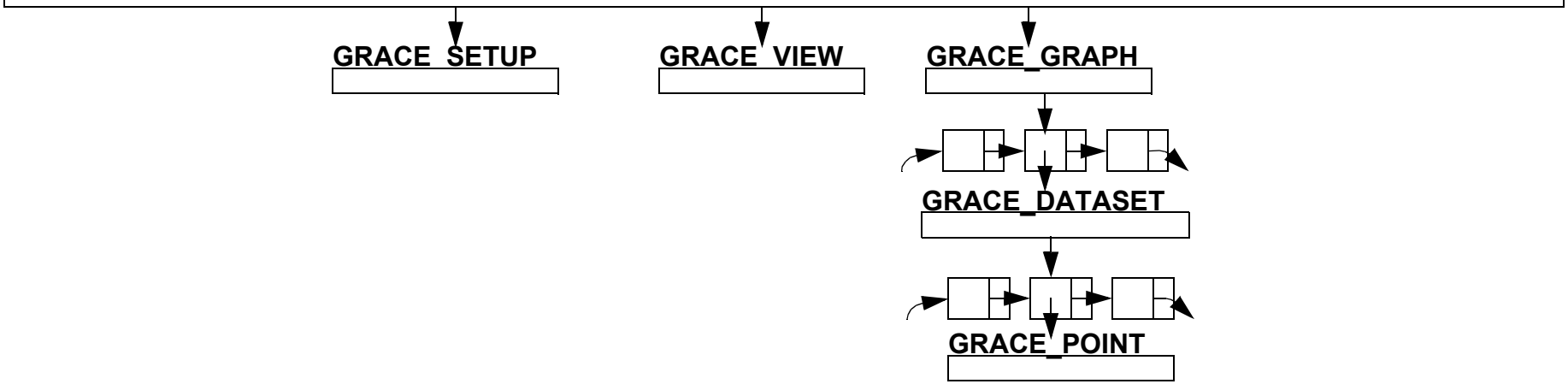
```
list_set(
    grace_graph->grace_dataset_list,
    grace_dataset );
```

GRACE_MULTIPLE

```
/* Usage */
/* Multiple datasets on single grace graph */
GRACE_MULTIPLE *grace_multiple_new(
    char *application_name,
    char *session_key,
    char *document_root_directory,
    char *title,
    char *sub_title,
    char *date_attribute_name,
    char *time_attribute_name,
    LIST *number_attribute_name_list,
    LIST *query_fetch_row_list,
    int horizontal_range_integer,
    double vertical_range );

/* Process */
GRACE_MULTIPLE *grace_multiple_calloc( void );

GRACE_SETUP *grace_setup =
    grace_setup_new(
        application_name,
        session_key,
        document_root_directory,
        title,
        sub_title,
        (char *)0 /* datatype_name */,
        1 /* grace_graph_list_length */ );

GRACE_VIEW *grace_view =
    grace_view_new(
        GRACE_VIEW_MIN_X,
        GRACE_VIEW_PORTRAIT_X_OFFSET,
        GRACE_VIEW_LANDSCAPE_X_OFFSET,
        1 /* grace_graph_list_length */,
        horizontal_range_integer,
        vertical_range,
        grace_page->landscape_boolean );

GRACE_GRAPH *grace_multiple_grace_graph(
    horizontal_range_integer,
    number_attribute_name_list
        /* dataset_name_list */,
    grace_view );

void grace_graph_query_fetch_row_list_set(
    grace_multiple_grace_graph(),
    date_attribute_name,
    time_attribute_name,
    number_attribute_name_list,
    query_fetch_row_list );

grace_multiple_grace_graph()->grace_world =
     grace_world_new(
          grace_multiple_grace_graph()->
               grace_dataset_list );

/* Usage */
GRACE_GRAPH *grace_multiple_grace_graph(
    int horizontal_range_integer,
    LIST *dataset_name_list,
    GRACE_VIEW *grace_view );

/* Process */
GRACE_GRAPH *grace_graph =
     grace_graph_new(
          0 /* graph_number */,
          1 /* xaxis_tick_on_boolean */,
          horizontal_range_integer,
          (char *) /* yaxis_heading */,
          grace_view );

for char *dataset_name in dataset_name_list
{
    list_set(
        grace_graph->grace_dataset_list,
        (GRACE_DATASET *
            grace_dataset =
                grace_dataset_new(
                    dataset_name,
                    list_new() /* grace_point_list */,
                    grace_dataset_xy,
                    grace_dataset_color_number(
                        GRACE_DATASET_MAX_COLOR_NUMBER,
                        1 /* cycle_colors_boolean */ ),
                    GRACE_DATASET_LINE_SYMBOL_SIZE ) ) );
}

/* Attributes */
GRACE_SETUP *grace_setup;
GRACE_VIEW *grace_view;
GRACE_GRAPH *grace_graph;
```

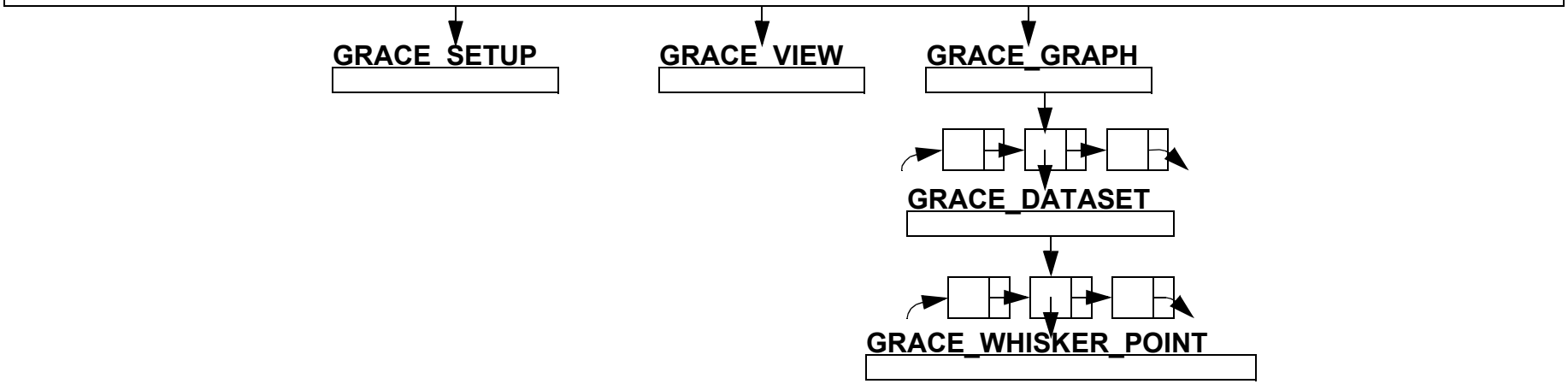
GRACE_GENERIC

```
/* Usage */
/* Interface with PROCESS_GENERIC_GRACE */
GRACE_GENERIC *grace_generic_new(
    char *application_name,
    char *appaserver_parameter->
        data_directory,
    char *value_attribute_name,
    char *process_generic_grace_title(),
    char *process_generic_grace_sub_title(),
    LIST *process_generic_grace_point_list(),
    int process_generic_grace_horizontal_range() );

/* Process */
GRACE_GENERIC *grace_generic_calloc( void );

GRACE_SETUP *grace_setup =
    grace_setup_new(
        application_name,
        (char *)0 /* session_key */,
        data_directory,
        process_generic_grace_title,
        process_generic_grace_sub_title,
        1 /* graph_list_length */ );

GRACE_VIEW *grace_view =
    grace_view_new(
        GRACE_VIEW_MIN_X,
        GRACE_VIEW_PORTRAIT_X_OFFSET,
        GRACE_VIEW_LANDSCAPE_X_OFFSET,
        1 /* graph_list_length */,
        process_generic_grace_horizontal_range,
        process_generic_grace_vertical_range,
        grace_setup->grace_page->landscape_boolean );

GRACE_GRAPH *grace_generic_grace_graph(
    value_attribute_name /* dataset_name */,
    process_generic_grace_point_list,
    process_generic_grace_horizontal_range,
    grace_view );

double grace_point_vertical_range_double(
    process_generic_grace_point_list() );

grace_generic_grace_graph()->grace_world =
      grace_world_new(
          grace_point_vertical_range_double(),
          grace_generic_grace_graph()->
              grace_dataset_list );

/* Usage */
GRACE_GRAPH *grace_generic_grace_graph(
    char *dataset_name,
    LIST *process_generic_grace_point_list(),
    int process_generic_grace_horizontal_range,
    GRACE_VIEW *grace_view );

/* Process */
GRACE_GRAPH *grace_graph =
      grace_graph_new(
          0 /* graph_number */,
          1 /* xaxis_tick_on_boolean */,
          process_generic_grace_horizontal_range,
          dataset_name /* yaxis_heading */,
          grace_view );

list_set(
    grace_graph->grace_dataset_list,
    ( GRACE_DATASET *grace_dataset =
        grace_dataset_new(
            dataset_name,
            process_generic_grace_point_list,
            grace_dataset_xy,
            grace_dataset_color_number(
                GRACE_DATASET_MAX_COLOR_NUMBER,
                0 /* not cycle_colors_boolean */ ),
            GRACE_DATASET_LINE_SYMBOL_SIZE ) ) );

/* Attributes */
GRACE_SETUP *grace_setup;
GRACE_VIEW *grace_view;
GRACE_GRAPH *grace_graph;
double grace_point_vertical_range_double;
```



GRACE_WHISKER

```
/* Usage */
/* Interface with PROCESS_GENERIC_WHISKER */
GRACE_WHISKER *grace_whisker_new(
    char *application_name,
    char *appaserver_parameter->
        data_directory,
    char *value_attribute_name,
    char *process_generic_whisker_title(),
    char *process_generic_whisker_sub_title(),
    LIST *process_generic_whisker_point_list(),
    int process_generic_whisker_horizontal_range(),
    double process_generic_whisker_vertical_range() );

/* Process */
GRACE_WHISKER *grace_whisker_calloc( void );

GRACE_SETUP *grace_setup =
    grace_setup_new(
        application_name,
        (char *)0 /* session_key */,
        data_directory,
        process_generic_whisker_title,
        process_generic_whisker_sub_title,
        1 /* graph_list_length */ );

GRACE_VIEW *grace_view =
    grace_view_new(
        GRACE_VIEW_MIN_X,
        GRACE_VIEW_PORTRAIT_X_OFFSET,
        GRACE_VIEW_LANDSCAPE_X_OFFSET,
        1 /* graph_list_length */,
        process_generic_whisker_horizontal_range,
        process_generic_whisker_vertical_range,
        grace_setup->grace_page->landscape_boolean );

GRACE_GRAPH *grace_whisker_grace_graph(
    value_attribute_name /* dataset_name */,
    process_generic_whisker_point_list,
    process_generic_whisker_horizontal_range,
    grace_view );

GRACE_WORLD *grace_whisker_grace_graph()->
    grace_world =
        grace_world_new(
            process_generic_whisker_vertical_range,
            grace_whisker_grace_graph()->
                grace_dataset_list );

/* Usage */
GRACE_GRAPH *grace_whisker_grace_graph(
    char *dataset_name,
    LIST *process_generic_whisker_point_list,
    int process_generic_whisker_horizontal_range,
    GRACE_VIEW *grace_view );

/* Process */
GRACE_GRAPH *grace_graph =
        grace_graph_new(
            0 /* graph_number */,
            1 /* xaxis_tick_on_boolean */,
            process_generic_whisker_horizontal_range,
            dataset_name /* yaxis_heading */,
            grace_view );

GRACE_DATASET *grace_dataset =
    grace_dataset_new(
        dataset_name,
        (LIST *)0 /* grace_point_list */,
        grace_dataset_whisker,
        grace_dataset_color_number(
            GRACE_DATASET_MAX_COLOR_NUMBER,
            0 /* not cycle_colors_boolean */ ),
        GRACE_DATASET_WHISKER_SYMBOL_SIZE );

grace_dataset->grace_whisker_point_list =
    process_generic_whisker_point_list;

list_set(
    grace_graph->grace_dataset_list,
    grace_dataset );

/* Attributes */
GRACE_SETUP *grace_setup;
GRACE_VIEW *grace_view;
GRACE_GRAPH *grace_graph;
```

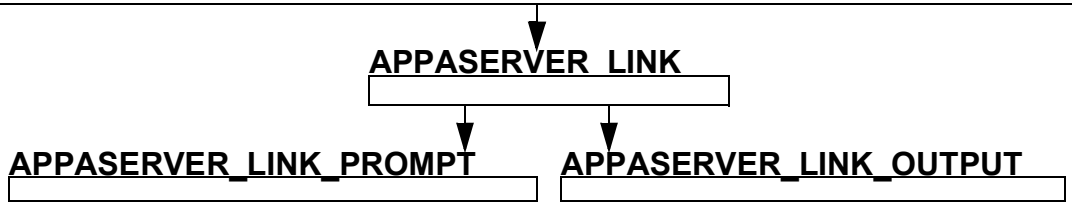


GRACE_SETUP

```
/* Usage */
GRACE_SETUP *grace_setup_new(
    char *application_name,
    char *session_key,
    char *application_parameter_data_directory,
    char *title,
    char *sub_title,
    char *chart_filename_key(),
    int grace_graph_list_length );

/* Process */
GRACE_SETUP *grace_setup_calloc( void );

GRACE_FILENAME *grace_filename =
    grace_filename_new(
        application_name,
        session_key,
        chart_filename_key,
        appaserver_parameter_data_directory );

char *grace_ps_system_string(
    GRACE_EXECUTABLE_TEMPLATE,
    grace_filename );

GRACE_PAGE *grace_page =
    grace_page_new(
        grace_graph_list_length );

char *grace_pdf_system_string(
    grace_filename );

GRACE_TITLE *grace_title =
    grace_title_new(
        title,
        sub_title );

/* Attributes */
GRACE_FILENAME *grace_filename;
char *grace_ps_system_string;
GRACE_PAGE *grace_page;
char *grace_pdf_system_string;
GRACE_TITLE *grace_title;
```

GRACE_FILENAME

```
/* Usage */
GRACE_FILENAME *grace_filename_new(
    char *application_name,
    char *session_key,
    char *filename_key,
    char *appaserver_parameter_data_directory() );

/* Process */
GRACE_FILENAME *grace_filename_calloc( void );

pid_t grace_filename_process_id(
    char *session_key );

APPASERVER_LINK *appaserver_link =
    appaserver_link_new(
        application_http_prefix(
            application_name ),
        application_server_address(),
        ( application_prepend_http_protocol_yn(
            application_name ) == 'y' ),
        appaserver_parameter_data_directory,
        GRACE_FILENAME_STEM,
        application_name,
        grace_filename_process_id(),
        session_key,
        filename_key /* begin_date_string */,
        (char *)0 /* end_date_string */,
        (char *)0 /* extension */ );

char *grace_filename_agr_output_filename(
    APPASERVER_LINK *appaserver_link );

char *grace_filename_agr_prompt_filename(
    APPASERVER_LINK *appaserver_link );

char *grace_filename_pdf_output_filename(
    APPASERVER_LINK *appaserver_link );

char *grace_filename_pdf_prompt_filename(
    APPASERVER_LINK *appaserver_link );

char *grace_filename_postscript_filename(
    APPASERVER_LINK *appaserver_link );

char *grace_home_directory =
    application_grace_home_directory(
        application_name );

char *grace_execution_directory =
    application_grace_execution_directory(
        application_name );

boolean grace_free_option =
    application_grace_free_option(
        application_name );

char *distill_directory =
    application_distill_directory(
        application_name );

char *ghost_script_directory =
    application_ghost_script_directory(
        application_name );

/* Attributes */
pid_t process_id;
APPASERVER_LINK *appaserver_link;
char *agr_output_filename;
char *agr_prompt_filename;
char *pdf_output_filename;
char *pdf_prompt_filename;
char *postscript_filename;
char *grace_home_directory;
char *grace_execution_directory;
boolean grace_free_option;
char *distill_directory;
char *ghost_script_directory;
```



GRACE_TITLE

```
/* Usage */
GRACE_TITLE *grace_title_new(
    char *title,
    char *sub_title );

/* Process */
GRACE_TITLE *grace_title_calloc( void );

int grace_title_length(
    char *title );

int title_char_size = GRACE_TITLE_CHAR_SIZE;

if ( grace_title_length() > 50 )
    title_char_size -= 0.2;

int sub_title_char_size =
    GRACE_TITLE_SUB_TITLE_CHAR_SIZE;

int grace_title_sub_title_length(
    char *sub_title );

if ( grace_title_sub_title_length() > 130 )
    sub_title_char_size -= 0.4;
else
if ( grace_title_sub_title_length() > 90 )
    sub_title_char_size -= 0.3;
else
if ( grace_title_sub_title_length() > 60 )
    sub_title_char_size -= 0.2;

/* Driver */
void grace_title_output(
    int GRACE_TITLE_MAX_SUB_TITLE_LENGTH,
    GRACE_TITLE *grace_title,
    FILE *output_pipe );

/* Attributes */
char *title;
char *sub_title;
int title_length;
int sub_title_length;
double title_char_size;
double sub_title_char_size;
```

GRACE_WORLD

```
/* Usage */
GRACE_WORLD *grace_world_new(
    LIST *grace_dataset_list );

/* Process */
GRACE_WORLD *grace_world_calloc(
    void );

double grace_dataset_minimum_horizontal(
    grace_dataset_list );

double grace_dataset_maximum_horizontal(
    grace_dataset_list );

double grace_world_horizontal_range(
    double grace_dataset_minimum_horizontal(),
    double grace_dataset_maximum_horizontal() );

double grace_world_min_x(
    double grace_dataset_minimum_horizontal() );

double grace_world_max_x(
    double grace_dataset_maximum_horizontal() );

double grace_dataset_minimum_vertical(
    grace_dataset_list );

double grace_dataset_maximum_vertical(
    grace_dataset_list );

double grace_world_min_y(
    grace_dataset_minimum_vertical() );

double grace_world_max_y(
    double grace_dataset_maximum_vertical() );

double grace_world_vertical_range(
    double grace_world_min_y(),
    double grace_dataset_maximum_vertical() );

double grace_world_y_tick_major(
    double grace_dataset_maximum_vertical(),
    double grace_world_max_y() );

int grace_world_y_tick_precision(
    double grace_world_y_tick_major() );

/* Usage */
double grace_world_min_y(
    double grace_dataset_minimum_vertical() );

/* Process */
if ( grace_dataset_minimum_vertical > 0.0 )
    return 0.0;
else
    return grace_dataset_minimum_vertical;

/* Usage */
int grace_world_y_tick_precision(
    double grace_world_y_tick_major() );

/* Process */
return
( grace_world_y_tick_major < 1.0 );

/* Usage */
void grace_world_output(
    GRACE_WORLD *grace_world,
    FILE *output_pipe );

/* Process */

/* Attributes */
double grace_dataset_minimum_horizontal;
double grace_dataset_maximum_horizontal;
double horizontal_range;
double min_x;
double max_x;
double grace_dataset_minimum_vertical;
double grace_dataset_maximum_vertical;
double vertical_range;
double min_y;
double max_y;
double y_tick_major;
int y_tick_precision;
```

GRACE_PAGE (1)

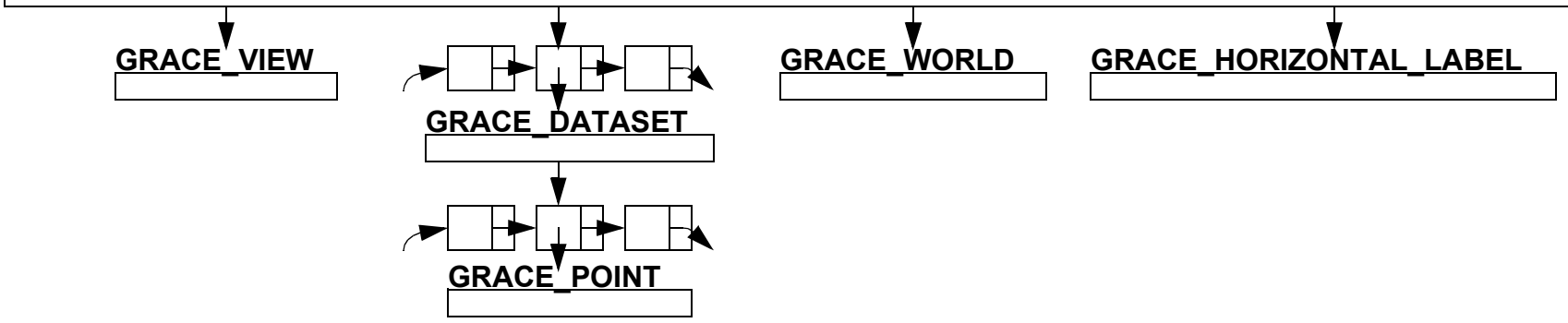
```
/* Usage */
GRACE_PAGE *grace_page_new(
    int grace_graph_list_length );

/* Process */
GRACE_PAGE *grace_page_calloc( void );

if (grace_graph_list_length <
    GRACE_PAGE_PORTRAIT_THRESHOLD )
{
    int width_pixels =
        GRACE_PAGE_LANDSCAPE_WIDTH_PIXELS;

    int length_pixels =
        GRACE_PAGE_LANDSCAPE_LENGTH_PIXELS;

    boolean landscape_boolean = 1;
}
else
{
    int width_pixels =
        GRACE_PAGE_PORTRAIT_WIDTH_PIXELS;

    int length_pixels =
        GRACE_PAGE_PORTRAIT_LENGTH_PIXELS;
}

/* Usage */
void grace_page_output(
    int width_pixels,
    int length_pixels,
    FILE *output_pipe );

/* Process */
char *grace_page_map_font_string();

char *grace_page_map_color_string();

char *grace_page_default_string();

char *grace_page_timestramp_string();

char *grace_page_region_string();

    fprintf(output_file,
"# Grace project file\n"
"# --------------------\n"
"# Appaserver generated\n"
"# --------------------\n"
"#\n"
"@version 50103\n"
"@page size %d, %d\n"
"@page scroll 5%c\n"
"@page inout 5%c\n"
"@link page off\n"
“%s\n”
“%s\n”
“%s\n”
“%s\n”
“%s\n”,
width_pixels,
length_pixels,
‘%’, ‘%’,
grace_page_map_font_string(),
grace_page_map_color_string(),
grace_page_default_string(),
grace_page_timestamp_string(),
grace_page_region_string() );


/* Attributes */
int width_pixels;
int length_pixels;
boolean landscape_boolean;
```

GRACE_PAGE (2)

```
/* Usage */
char *grace_page_map_font_string( void );

/* Process */
return
"@map font 0 to \"Times-Roman\", \"Times-Roman\"\n"
"@map font 1 to \"Times-Italic\", \"Times-Italic\"\n"
"@map font 2 to \"Times-Bold\", \"Times-Bold\"\n"
"@map font 3 to \"Times-BoldItalic\", \"Times-BoldItalic\"\n"
"@map font 4 to \"Helvetica\", \"Helvetica\"\n"
"@map font 5 to \"Helvetica-Oblique\", \"Helvetica-Oblique\"\n"
"@map font 6 to \"Helvetica-Bold\", \"Helvetica-Bold\"\n"
"@map font 7 to \"Helvetica-BoldOblique\", \"Helvetica-BoldOblique\"\n"
"@map font 8 to \"Courier\", \"Courier\"\n"
"@map font 9 to \"Courier-Oblique\", \"Courier-Oblique\"\n"
"@map font 10 to \"Courier-Bold\", \"Courier-Bold\"\n"
"@map font 11 to \"Courier-BoldOblique\", \"Courier-BoldOblique\"\n"
"@map font 12 to \"Symbol\", \"Symbol\"\n"
"@map font 13 to \"ZapfDingbats\", \"ZapfDingbats\"\n";

/* Usage */
char *grace_page_map_color_string( void );

/* Process */
return
"@map color 0 to (255, 255, 255), \"white\"\n"
"@map color 1 to (0, 0, 0), \"black\"\n"
"@map color 2 to (255, 0, 0), \"red\"\n"
"@map color 3 to (0, 0, 255), \"blue\"\n"
"@map color 4 to (0, 255, 0), \"green\"\n"
"@map color 5 to (255, 165, 0), \"orange\"\n"
"@map color 6 to (188, 143, 143), \"brown\"\n"
"@map color 7 to (148, 0, 211), \"violet\"\n"
"@map color 8 to (0, 255, 255), \"cyan\"\n"
"@map color 9 to (255, 0, 255), \"magenta\"\n"
"@map color 10 to (103, 7, 72), \"maroon\"\n"
"@map color 11 to (114, 33, 188), \"indigo\"\n"
"@map color 12 to (0, 125, 255), \"gr_blue\"\n"
"@map color 13 to (64, 224, 208), \"turquoise\"\n"
"@map color 14 to (0, 139, 0), \"green4\"\n"
"@map color 15 to (220, 220, 220), \"grey\"\n";

/* Usage */
char *grace_page_default_string( void );

/* Process */
sprintf( static default_string[],
"@default linewidth 1.0\n"
"@default linestyle 1\n"
"@default color 1\n"
"@default pattern 1\n"
"@default font 6\n"
"@default char size 1.000000\n"
"@default symbol size 1.000000\n"
"@default sformat \"%c16.8f\"\n"
"@reference date 0\n"
"@date wrap on\n"
"@date wrap year 1900\n"
"@background color 0\n"
"@page background fill on\n",
'%' );

/* Usage */
char *grace_page_timestamp_string( void );

/* Process */
return
"@timestamp off\n"
"@timestamp 0.03, 0.03\n"
"@timestamp color 1\n"
"@timestamp rot 0\n"
"@timestamp font 6\n"
"@timestamp char size 1.000000\n";

/* Usage */
char *grace_page_region_string( void );

/* Process */
static char region_string[];
char *ptr = region_string;

for ( int region = 0; region < 5; region++ )
{
    ptr += sprintf( ptr,
        "@r%d type above\n"
        "@r%d linestyle 1\n"
        "@r%d linewidth 1.0\n"
        "@r%d color 1\n"
        "@r%d line 0, 0, 0, 0\n",
        region,
        region
        region,
        region,
        region );
}
```

GRACE_GRAPH (1)

```
/* Usage */
GRACE_GRAPH *grace_graph_new(
    int graph_number,
    boolean xaxis_tick_on_boolean,
    int horizontal_range_integer,
    char *yaxis_heading,
    GRACE_VIEW *grace_view );

/* Process */
GRACE_GRAPH *grace_graph_calloc(
    void );

LIST *grace_dataset_list = list_new();

char *grace_graph_xaxis_tick_on_off(
    boolean xaxis_tick_on_boolean );

char *grace_graph_xaxis_ticklabel_format(
    int horizontal_range_integer );

int grace_graph_xaxis_ticklabel_angle(
    int horizontal_range_integer );

/* Usage */
void grace_graph_list_query_fetch_row_list_set(
    LIST *grace_graph_list,
    char *date_attribute_name,
    char *time_attribute_name,
    LIST *number_attribute_name_list,
    LIST *query_fetch_row_list );

/* Process */
for QUERY_ROW *query_row in
    query_fetch_row_list
{
    QUERY_CELL *date_query_cell =
        query_cell_seek(
            date_attribute_name,
            query_row->cell_list );

    if ( time_attribute_name )
    {
        QUERY_CELL *time_query_cell =
            query_cell_seek(
                time_attribute_name,
                query_row->cell_list );
    }

    double grace_graph_cell_horizontal_double(
        date_query_cell,
        time_query_cell );

    if ( !grace_graph_cell_horizontal_double() )
            continue;

    for char *number_attribute_name in
        number_attribut_name_list
    {
        GRACE_DATASET *grace_dataset =
            grace_dataset_graph_list_seek(
                number_attribute_name
                    /* datatype_name */,
                grace_graph_list );

        QUERY_CELL *number_query_cell =
            query_cell_seek(
                number_attribute_name,
                query_row->cell_list );

        boolean grace_graph_null_boolean(
            char *number_query_cell->select_datum );

        if ( !grace_graph_null_boolean() )
        {
            double grace_graph_vertical_double(
                char *number_query_cell->
                    select_datum );

            if ( !grace_dataset->point_list )
                grace_dataset->point_list =
                    list_new();

            list_set(
                grace_dataset->grace_point_list,
                grace_point_new(
                    grace_graph_cell_horizontal_double(),
                    grace_graph_vertical_double() ) );
        }
    }

/* Attributes */
int graph_number;
char *yaxis_heading;
GRACE_VIEW *grace_view;
LIST *grace_dataset_list /* initialized only */;
char *xaxis_tick_on_off;
char *xaxis_ticklabel_format;
int xaxis_ticklabel_angle;

/* Set externally */
GRACE_WORLD *grace_world;
GRACE_HORIZONTAL_LABEL *grace_horizontal_label;
```



GRACE_GRAPH (2)

```
/* Usage */
void grace_graph_query_fetch_row_list_set(
    GRACE_GRAPH *grace_graph,
    char *date_attribute_name,
    char *time_attribute_name,
    LIST *number_attribute_name_list,
    LIST *query_fetch_row_list );

/* Process */
for QUERY_ROW *query_row in
    query_fetch_row_list
{
    QUERY_CELL *date_query_cell =
        query_cell_seek(
            date_attribute_name,
            query_row->cell_list );

    if ( time_attribute_name )
    {
        QUERY_CELL *time_query_cell =
            query_cell_seek(
                time_attribute_name,
                query_row->cell_list );
    }

    double grace_graph_cell_horizontal_double(
        date_query_cell,
        time_query_cell );

    if ( !grace_graph_cell_horizontal_double() )
        continue;

    for char *number_attribute_name in
        number_attribut_name_list
    {
        GRACE_DATASET *grace_dataset =
            grace_dataset_seek(
                number_attribute_name
                    /* datatype_name */,
                grace_graph->grace_dataset_list );

        QUERY_CELL *number_query_cell =
            query_cell_seek(
                number_attribute_name,
                query_row->cell_list );

        boolean grace_graph_null_boolean(
            number_query_cell->select_datum );

        if ( !grace_graph_null_boolean() )
        {
            double grace_graph_vertical_double(
                number_query_cell->select_datum );

            if ( !grace_dataset->grace_point_list )
                grace_dataset->grace_point_list =
                    list_new();

            list_set(
                grace_dataset->grace_point_list,
                grace_point_new(
                    grace_graph_cell_horizontal_double(),
                    grace_graph_vertical_double() ) );
        }
    }
```

GRACE_GRAPH (3)

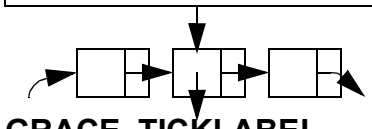
```
/* Usage */
double grace_graph_cell_horizontal_double(
    QUERY_CELL *date_query_cell,
    QUERY_CELL *time_query_cell );

/* Process */
double grace_graph_horizontal_double(
    char *date_string,
    char *time_string );

/* Driver */
void grace_graph_output(
    GRACE_TITLE *grace_title,
    GRACE_GRAPH *grace_graph,
    FILE *output_file );

/* Process */
void grace_graph_top_output(
    int grace_graph->graph_number,
    FILE *output_file );

void grace_world_output(
    grace_graph->grace_world,
    output_file );

void grace_view_output(
    grace_graph->grace_view,
    output_file );

if ( grace_title )
{
    grace_title_output(
        GRACE_TITLE_MAX_SUB_TITLE_LENGTH,
        grace_title,
        output_file );
}

if ( grace_graph->grace_horizonatal_label )
{
    grace_horizontal_label_output(
        grace_graph->grace_horizontal_label,
        output_file );
}

void fprintf( output_file,
    "@    xaxes scale Normal\n"
    "@    yaxes scale Normal\n"
    "@    xaxes invert off\n"
    "@    yaxes invert off\n" );

void grace_graph_xaxis_output(
    char *grace_graph->xaxis_tick_on_off,
    double grace_graph->grace_view->x_tick_major,
    int grace_graph->grace_view->x_tick_minor,
    char *grace_graph->xaxis_ticklabel_format,
    int xaxis_ticklabel_angle,
    FILE *output_file );

void grace_graph_yaxis_output(
    double grace_graph->grace_world->y_tick_major,
    int grace_graph->grace_world->y_tick_precision,
    char *grace_graph->yaxis_heading,
    FILE *output_file );

void grace_dataset_list_output(
    grace_graph->graph_number,
    grace_graph->grace_dataset_list,
    output_file );
```

GRACE_HORIZONTAL_LABEL

```
/* Usage */
GRACE_HORIZONTAL_LABEL *
   grace_horizontal_label_new(
      void );

/* Process */
GRACE_HORIZONTAL_LABEL *
   grace_horizontal_label_calloc(
      void );

LIST *grace_ticklabel_list = list_new();

/* Usage */
void grace_horizontal_label_output(
   GRACE_HORIZONTAL_LABEL *grace_horizontal_label,
   FILE *output_file );
```

```
/* Process */
double grace_ticklabel_first_horizontal_double(
   grace_horizontal_label->
      grace_ticklabel_list );

int grace_ticklabel_list_length =
   list_length(
      grace_horizontal_label->
         grace_ticklabel_list );

char *grace_horizontal_label_agr_string(
   double grace_ticklabel_first_horizontal_double(),
   int grace_ticklabel_list_length );

fprintf(     output_file,
             "%s\n",
             grace_horizontal_label_agr_string() );
```

```
void grace_ticklabel_list_output(
   grace_horizontal_label->
      grace_ticklabel_list,
      output_file );

char *grace_horizontal_label_tick_minor_string(
   int grace_ticklabel_list_length );

fprintf(     output_file,
             "%s\n",
             grace_horizontal_label_tick_minor_string() );

/* Attributes */
LIST *grace_ticklabel_list;
```



GRACE_TICKLABEL

```
/* Usage */
GRACE_TICKLABEL *grace_ticklabel_new(
   double horizontal_double,
   char *ticklabel_string );

/* Process */
GRACE_TICKLABEL *grace_ticklabel_calloc( void );

int grace_ticklabel_major_integer( void );

char *grace_ticklabel_agr_string(
   int grace_ticklabel_major_integer(),
   double horizontal_double,
   char *ticklabel_string );
```

```
/* Usage */
double grace_ticklabel_first_horizontal_double(
   LIST *grace_ticklabel_list );

/* Process */
GRACE_TICKLABEL *grace_ticklabel =
   list_first(
      grace_ticklabel_list );

return grace_ticklabel->horizontal_double;

/* Usage */
void grace_ticklabel_list_output(
   LIST *grace_ticklabel_list,
   FILE *output_file );
```

```
/* Process */
for GRACE_TICKLABEL *grace_ticklabel in
   grace_ticklabel_list
{
   fprintf(     output_file,
                "%s\n",
                grace_ticklabel->agr_string );
}

/* Attribute */
double horizontal_double;
char *ticklabel_string;
char *agr_string;
```

GRACE_VIEW (1)

```
/* Usage */
GRACE_VIEW *grace_view_new(
    double GRACE_VIEW_MIN_X,
    double GRACE_VIEW_PORTRAIT_X_OFFSET,
    double GRACE_VIEW_LANDSCAPE_X_OFFSET,
    int grace_graph_list_length,
    int horizontal_range_integer,
    boolean landscape_boolean );

/* Process */
GRACE_VIEW *grace_view_calloc(
    void );

double grace_view_max_x(
    double GRACE_VIEW_MIN_X,
    double GRACE_VIEW_PORTRAIT_X_OFFSET,
    double GRACE_VIEW_LANDSCAPE_X_OFFSET,
    boolean landscape_boolean );

double grace_view_min_y(
    grace_graph_list_length );

double grace_view_y_offset(
    grace_graph_list_length,
    landscape_boolean );

double grace_view_max_y(
    double grace_view_min_y(),
    double grace_view_y_offset() );

double grace_view_y_increment(
    grace_graph_list_length,
    landscape_boolean );

double grace_view_x_tick_major(
    horizontal_range_integer );

int grace_view_x_tick_minor(
    horizontal_range_integer );

/* Usage */
double grace_view_min_y(
    int grace_graph_list_length );

/* Process */
if ( grace_graph_list_length == 1 )
    min_y = 0.7;
else
    min_y = 0.05;
}

/* Driver */
void grace_view_output(
    GRACE_VIEW *grace_view,
    FILE *output_pipe );

/* Attributes */
double min_x;
double max_x;
double min_y;
double y_offset;
double max_y;
double y_increment;
double x_tick_major;
int x_tick_minor;
```

GRACE_VIEW (2)

```
/* Usage */
double grace_view_y_increment(
    int grace_graph_list_length,
    boolean landscape_boolean );

/* Process */
if ( landscape_boolean )
{
    if ( grace_graph_list_length == 1 )
        y_increment = 0.0;
    else
    if ( grace_graph_list_length == 2 )
        y_increment = 0.44;
    else
    if ( grace_graph_list_length == 3 )
        y_increment = 0.3;
    else
    if ( grace_graph_list_length == 4 )
        y_increment = 0.23;
    else
    if ( grace_graph_list_length == 5 )
        y_increment = 0.175;
    else
        y_increment =
            0.75 / (double)grace_graph_list_length;
}
else
{
    if ( grace_graph_list_length == 1 )
        y_increment = 0.0;
    else
    if ( grace_graph_list_length == 2 )
        y_increment = 0.53;
    else
    if ( grace_graph_list_length == 3 )
        y_increment = 0.41;
    else
    if ( grace_graph_list_length == 4 )
        y_increment = 0.3;
    else
    if ( grace_graph_list_length == 5 )
        y_increment = 0.24;
    else
    if ( grace_graph_list_length == 6 )
        y_increment = 0.2;
    else
    if ( grace_graph_list_length == 7 )
        y_increment = 0.17;
    else
    if ( grace_graph_list_length == 8 )
        y_increment = 0.15;
    else
    if ( grace_graph_list_length == 9 )
        y_increment = 0.13;
    else
    if ( grace_graph_list_length == 10 )
        y_increment = 0.12;
    else
    if ( grace_graph_list_length == 11 )
        y_increment = 0.07;
    else
    if ( grace_graph_list_length == 12 )
        y_increment = 0.05;
    else
        y_increment =
            0.6 / (double)grace_graph_list_length;
}

/* Usage */
double grace_view_y_offset(
    int grace_graph_list_length,
    boolean landscape_boolean );

/* Process */
if ( landscape_boolean )
{
    if ( grace_graph_list_length == 1 )
        y_offset = 0.46;
    else
    if ( grace_graph_list_length == 2 )
        y_offset = 0.375;
    else
    if ( grace_graph_list_length == 3 )
        y_offset = 0.23;
    else
    if ( grace_graph_list_length == 4 )
        y_offset = 0.16;
    else
    if ( grace_graph_list_length == 5 )
        y_offset = 0.12;
    else
        y_offset =
            0.6 / (double)grace_graph_list_length;
}
else
{
    if ( grace_graph_list_length == 1 )
        y_offset = 0.46;
    else
    if ( grace_graph_list_length == 2 )
        y_offset = 0.4;
    else
    if ( grace_graph_list_length == 3 )
        y_offset = 0.31;
    else
    if ( grace_graph_list_length == 4 )
        y_offset = 0.23;
    else
    if ( grace_graph_list_length == 5 )
        y_offset = 0.17;
    else
    if ( grace_graph_list_length == 6 )
        y_offset = 0.14;
    else
    if ( grace_graph_list_length == 7 )
        y_offset = 0.11;
    else
    if ( grace_graph_list_length == 8 )
        y_offset = 0.09;
    else
    if ( grace_graph_list_length == 9 )
        y_offset = 0.08;
    else
    if ( grace_graph_list_length == 10 )
        y_offset = 0.075;
    else
    if ( grace_graph_list_length == 11 )
        y_offset = 0.07;
    else
    if ( grace_graph_list_length == 12 )
        y_offset = 0.065;
    else
        y_offset =
            0.8 / (double)grace_graph_list_length;
}
```

GRACE_VIEW (3)

```
/* Usage */
double grace_view_x_tick_major(
    int horizontal_range_integer );

/* Process */
if ( !horizontal_range_integer )
    x_tick_major = 0.0;
else
if ( horizontal_range_integer <= 1 )
    x_tick_major = 0.25;
else
if ( horizontal_range_integer == 2 )
    x_tick_major = 0.5;
else
if ( horizontal_range_integer <= 7 )
    x_tick_major = 1.0;
else
if ( horizontal_range_integer <= 14 )
    x_tick_major = 2.0;
else
if ( horizontal_range_integer <= 62 )
    x_tick_major = 7.0;
else
if ( horizontal_range_integer <= 100 )
    x_tick_major = 10.0;
else
if ( horizontal_range_integer <= 180 )
    x_tick_major = 15.0;
else
    x_tick_major =
        (double)horizontal_range_integer / 25.0;

/* Usage */
int grace_view_x_tick_minor(
    int horizontal_range_integer );

/* Process */
if ( horizontal_range_integer <= 1 )
    x_tick_minor = 5;
else
if ( horizontal_range_integer == 2 )
    x_tick_minor = 4;
else
if ( horizontal_range_integer <= 7 )
    x_tick_minor = 3;
else
if ( horizontal_range_integer <= 14 )
    x_tick_minor = 2;
else
if ( horizontal_range_integer <= 62 )
    x_tick_minor = 1;
else
    x_tick_minor = 0;

/* Usage */
GRACE_VIEW *grace_view_increment(
    GRACE_VIEW *grace_view );

/* Process */
GRACE_VIEW *grace_view_next =
    grace_view_calloc();

memcpy(
    grace_view_next,
    grace_view,
    sizeof ( GRACE_VIEW ) );

min_y += y_increment;

double grace_view_max_y(
    min_y,
    y_offset );
```

GRACE_DATASET (1)

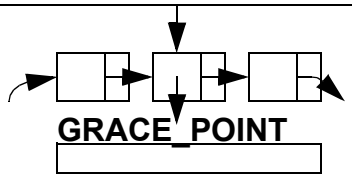
```
/* Usage */
GRACE_DATASET *grace_dataset_new(
    char *datatype_name,
    LIST *grace_point_list,
    enum grace_dataset_enum grace_dataset_enum,
    int grace_dataset_color_number(),
    double symbol_size );

/* Process */
GRACE_DATASET *grace_dataset_calloc( void );

char *grace_dataset_legend(
    char *datatype_name );

int grace_dataset_number( void );

int grace_dataset_line_type(
    enum grace_dataset_enum
        grace_dataset_enum );

char *grace_dataset_graph_type(
    enum grace_dataset_enum
        grace_dataset_enum );

/* Usage */
GRACE_DATASET *grace_dataset_graph_list_seek(
    char *datatype_name,
    LIST *grace_graph_list );

/* Process */
for GRACE_GRAPH *grace_graph in grace_graph_list
{
    GRACE_DATASET *grace_dataset_seek(
        datatype_name,
        grace_graph->grace_dataset_list );

    if ( grace_dataset_seek() )
        return grace_dataset_seek();
}
return (GRACE_DATASET *)0;
```

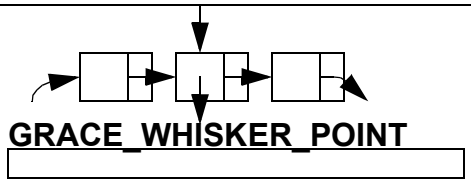
```
/* Usage */
GRACE_DATASET *grace_dataset_seek(
    char *datatype_name,
    LIST *grace_dataset_list );

/* Process */

/* Usage */
double grace_dataset_minimum_horizontal(
    LIST *grace_dataset_list );

/* Process */
if ( grace_dataset->
        grace_dataset_enum ==
            grace_dataset_whisker )
{
    double grace_whisker_point_minimum_horizontal(
        grace_dataset->grace_whisker_point_list );
}
else
{
    double grace_point_minimum_horizontal(
        grace_dataset->grace_point_list );
}

/* Usage */
double grace_dataset_maximum_horizontal(
    LIST *grace_dataset_list );

/* Process */
if ( grace_dataset->
        grace_dataset_enum ==
            grace_dataset_whisker )
{
    double grace_whisker_point_maximum_horizontal(
        grace_dataset->grace_whisker_point_list );
}
else
{
    double grace_point_maximum_horizontal(
        grace_dataset->grace_point_list );
}

/* Usage */
double grace_dataset_minimum_vertical(
    LIST *grace_dataset_list );

/* Process */
if ( grace_dataset->
        grace_dataset_enum ==
            grace_dataset_whisker )
{
    double grace_whisker_point_minimum_vertical(
        grace_dataset->grace_whisker_point_list );
}
else
{
    double grace_point_minimum_vertical(
        grace_dataset->grace_point_list );
}

/* Usage */
double grace_dataset_maximum_vertical(
    LIST *grace_dataset_list );

/* Process */
if ( grace_dataset->
        grace_dataset_enum ==
            grace_dataset_whisker )
{
    double grace_whisker_point_maximum_vertical(
        grace_dataset->grace_whisker_point_list );
}
else
{
    double grace_point_maximum_vertical(
        grace_dataset->grace_point_list );
}
```

GRACE_DATASET (2)

```
/* Usage */
int grace_dataset_color_number(
    int GRACE_DATASET_MAX_COLOR_NUMBER,
    boolean cycle_colors_boolean );

/* Process */

/* Usage */
LIST *grace_dataset_point_list(
    char *date_attribute_name,
    char *time_attribute_name,
    LIST *query_fetch_row_list,
    char *grace_datatype_name );

/* Process */
LIST *grace_point_list = list_new();

for QUERY_ROW *query_row in query_fetch_row_list
{
    QUERY_CELL *number_query_cell =
        query_cell_seek(
            grace_datatype_name,
            query_row->cell_list );

    boolean grace_graph_null_boolean(
        number_query_cell->select_datum );

    if ( grace_graph_null_boolean() ) continue;

    double grace_graph_vertical_double(
        number_query_cell->
            select_datum );

    QUERY_CELL *date_query_cell =
        query_cell_seek(
            date_attribute_name,
            query_row->cell_list );

    if ( time_attribute_name )
    {
        QUERY_CELL *time_query_cell =
            query_cell_seek(
                time_attribute_name,
                query_row->cell_list );
    }

    double grace_graph_cell_horizontal_double(
        date_query_cell,
        time_query_cell );

    GRACE_POINT *grace_point =
        grace_point_new(
            grace_graph_cell_horizontal_double(),
            grace_graph_vertical_double() );

    list_set(
        grace_point_list,
        grace_point );
}

/* Usage */
void grace_dataset_list_output(
    int grace_graph->graph_number,
    LIST *grace_graph->grace_dataset_list,
    FILE *output_file );

/* Process */
for GRACE_DATASET *grace_dataset in
    grace_dataset_list
{
    if ( grace_dataset->
            grace_dataset_enum ==
                grace_dataset_whisker
    && !list_length(
            grace_dataset->
                grace_whisker_point_list ) )
    {
        continue;
    }
    else
    if ( !list_length(
            grace_dataset->
                grace_point_list ) )
    {
        continue;
    }

    void grace_dataset_output(
        int graph_number,
        int grace_dataset->color_number,
        char *grace_dataset->legend,
        int grace_dataset->dataset_number,
        int grace_dataset->line_type,
        char *grace_dataset->graph_type,
        double grace_dataset->symbol_size,
        FILE *output_file );

    if ( grace_dataset->
            grace_dataset_enum ==
                grace_dataset_whisker )
    {
        void grace_whisker_point_list_output(
            grace_dataset->grace_whisker_point_list,
            output_file );
    }
    else
    {
        void grace_point_list_output(
            grace_dataset->grace_point_list,
            output_file );
    }
}

/* Attributes */
char *datatype_name;
enum grace_dataset_enum grace_dataset_enum;
int color_number;
LIST *grace_point_list;
LIST *grace_whisker_point_list;
char *legend;
int dataset_number;
int line_type;
char *graph_type;
double symbol_size;
```

grace_dataset_xy
grace_dataset_bar
grace_dataset_whisker





GRACE_POINT (1)

```
/* Usage */
GRACE_POINT *grace_point_new(
    double horizontal_double,
    double vertical_double );

/* Process */
GRACE_POINT *grace_point_calloc( void );

/* Usage */
double grace_point_minimum_horizontal(
    LIST *grace_point_list );

/* Process */
GRACE_POINT *grace_point =
    list_first( grace_point_list );

grace_point->horizontal_double;

/* Usage */
double grace_point_maximum_horizontal(
    LIST *grace_point_list );

/* Process */
GRACE_POINT *grace_point =
    list_last( grace_point_list );

grace_point->horizontal_double;

/* Usage */
double grace_point_minimum_vertical(
    LIST *grace_point_list );

/* Process */
double minimum_vertical = FLOAT_MAXIMUM_DOUBLE;

for GRACE_POINT *grace_point in grace_point_list
{
    if (    grace_point->vertical_double <
            minimum_vertical )
    {
        minimum_vertical = grace_point->vertical_double;
    }
}

/* Usage */
double grace_point_maximum_vertical_double(
    LIST *grace_point_list );

/* Process */
double maximum_vertical = FLOAT_MINIMUM_DOUBLE;

for GRACE_POINT *grace_point in grace_point_list
{
    if (    grace_point->vertical_double >
            maximum_vertical )
    {
        maximum_vertical =
            grace_point->vertical_double;
    }
}

/* Usage */
GRACE_POINT *grace_point_delimited_parse(
    char delimiter,
    int date_piece,
    int time_piece,
    int value_piece,
    char *delimited_row );

/* Process */
char *grace_point_date_string(
    char delimiter,
    int date_piece,
    char *delimited_row );

if ( time_piece > -1 )
{
    char *grace_point_time_string(
        char delimiter,
        int time_piece,
        char *delimited_row );
}

double grace_graph_horizontal_double(
    grace_point_date_string(),
    grace_point_time_string() );

if ( !grace_graph_horizontal_double() ) return NULL;

boolean grace_point_value_null_boolean(
    char delimiter,
    int value_piece,
    char *delimited_row );

if ( grace_point_value_null_boolean() ) return NULL;

double grace_point_value_double(
    char delimiter,
    int value_piece,
    char *delimited_row );

GRACE_POINT *grace_point_new(
    grace_point_horizontal_double(),
    grace_point_value_double()
        /* vertical_double */ );

/* Driver */
void grace_point_list_output(
    LIST *grace_dataset->grace_point_list,
    FILE *output_file );

/* Process */
for GRACE_POINT *grace_point in grace_point_list
{
    void grace_point_output(
        GRACE_POINT *grace_point,
        FILE *output_file );
}

fprintf( output_file, "&\n" );

/* Attributes */
double horizontal_double;
double vertical_double;
```

GRACE_POINT (2)

```
/* Usage */
double grace_point_horizontal_range_double(
     LIST *grace_point_list );

/* Process */
double minimum_horizontal =
   FLOAT_MAXIMUM_DOUBLE;

double maximum_horizontal =
   FLOAT_MINIMUM_DOUBLE;

for GRACE_POINT *grace_point in grace_point_list
{
   if (   grace_point->horizontal_double <
          minimum_horizontal )
   {
      minimum_horizontal =
         grace_point->horizontal_double;
   }
   else
   if (   grace_point->horizontal_double >
          maximum_horizontal )
   {
      maximum_horizonal =
         grace_point->horizontal_double;
   }
}

return
maximum_horizontal - minimum_horizontal;

/* Usage */
double grace_point_vertical_range_double(
     LIST *grace_point_list );

/* Process */
double minimum_vertical = FLOAT_MAXIMUM_DOUBLE;
double maximum_vertical = FLOAT_MINIMUM_DOUBLE;

for GRACE_POINT *grace_point in grace_point_list
{
   if (   grace_point->vertical_double <
          minimum_vertical )
   {
      minimum_vertical = grace_point->vertical_double;
   }
   else
   if (   grace_point->vertical_double >
          maximum_vertical )
   {
      maximum_vertical =
         grace_point->vertical_double;
   }
}

return
maximum_vertical - minimum_vertical;
```

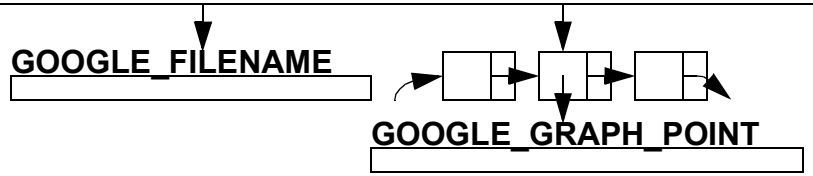
GRACE_WHISKER_POINT

```
/* Usage */
GRACE_WHISKER_POINT *grace_whisker_point_new(
    double horizontal_double,
    double low_value,
    double average_value,
    double high_value );

/* Process */
GRACE_WHISKER_POINT *
    grace_whisker_point_calloc(
        void );

/* Usage */
double grace_whisker_point_minimum_horizontal(
        LIST *grace_whisker_point_list );

/* Process */
GRACE_WHISKER_POINT *grace_whisker_point =
    list_first( grace_whisker_point_list );

grace_whisker_point->horizontal_double;

/* Usage */
double grace_whisker_point_maximum_horizontal(
        LIST *grace_whisker_point_list );

/* Process */
GRACE_WHISKER_POINT *grace_whisker_point =
    list_last( grace_whisker_point_list );

grace_whisker_point->horizontal_double;

/* Usage */
double grace_whisker_point_minimum_vertical(
    LIST *grace_whisker_point_list );

/* Process */

/* Usage */
double grace_whisker_point_maximum_vertical(
    LIST *grace_whisker_point_list );

/* Process */
```

```
/* Usage */
GRACE_WHISKER_POINT *
   grace_whisker_point_delimited_parse(
       char delimiter,
       int date_piece,
       int time_piece,
       int average_value_piece,
       int low_value_piece,
       int high_value_piece,
       char *delimited_row );

/* Process */
char *grace_point_date_string(
   delimiter,
   date_piece,
   delimited_row );

if ( time_piece > -1 )
{
   char *grace_point_time_string(
       delimiter,
       time_piece,
       delimited_row );
}

double grace_graph_horizontal_double(
   grace_point_date_string(),
   grace_point_time_string() );

if ( !grace_graph_horizontal_double() )
       return (GRACE_WHISKER_POINT *)0;

boolean grace_point_value_null_boolean(
   delimiter,
   average_value_piece,
   delimited_row );

if ( grace_point_value_null_boolean() )
       return (GRACE_WHISKER_POINT *)0;

double average_value =
   grace_point_value_double(
       delimiter,
       average_value_piece,
       delimited_row );

double low_value =
   grace_point_value_double(
       delimiter,
       low_value_piece,
       delimited_row );

double high_value =
   grace_point_value_double(
       delimiter,
       high_value_piece,
       delimited_row );

GRACE_WHISKER_POINT *
   grace_whisker_point_new(
       grace_point_horizontal_double(),
       low_value,
       average_value,
       high_value );

/* Driver */
void grace_whisker_point_list_output(
     LIST *grace_dataset->
          grace_whisker_point_list,
     FILE *output_file );

/* Process */
for GRACE_WHISKER_POINT *
     grace_whisker_point in
          grace_whisker_point_list
{
     void grace_whisker_point_output(
          GRACE_WHISKER_POINT *grace_whisker_point,
          FILE *output_file );
}

fprintf( output_file, "&\n" );

/* Attributes */
double horizontal_double;
double low_value;
double average_value;
double high_value;
```

GOOGLE_GRAPH (1)

```
/* Usage */
GOOGLE_GRAPH *google_graph_new(
    char *application_name,
    char *session_key,
    char *data_directory,
    char *date_attribute_name,
    char *time_attribute_name,
    char *number_attribute_name,
    LIST *query_fetch_row_list,
    char *chart_title(),
    char *chart_filename_key(),
    int chart_number,
    char *google_graph_function
        /* either LineChart or ColumnChart */ );

/* Process */
GOOGLE_GRAPH *google_graph_calloc(
    void );

GOOGLE_FILENAME *google_filename =
    google_filename_new(
        application_name,
        session_key,
        data_directory,
        (char *)0 /* number_attribute_name */,
        chart_filename_key );

char *google_graph_visualization_function_name(
    chart_number );

char *google_graph_id_string(
    chart_number );

char *google_graph_loader_script_html();

char *google_graph_corechart_script_html();

char *google_graph_function_script_open(
    google_graph_visualization_function_name() );

char *google_graph_number_attribute_label(
    char *number_attribute_name );

char *google_graph_datatable_statement(
    google_graph_number_attribute_label() );

char *google_graph_options_JSON(
    chart_title );

char *google_graph_instantiation_statement(
    google_graph_id_string(),
    google_graph_function );

char *google_graph_draw_statement();

LIST *google_graph_point_list(
    date_attribute_name,
    time_attribute_name,
    number_attribute_name,
    query_fetch_row_list );

char *google_graph_function_script_close();

char *google_graph_set_callback_script_html(
    char *google_graph_visualization_function_name() );

char *google_graph_division_tag(
    google_graph_id_string() );

char *google_graph_document_open_html(
    google_graph_loader_script_html(),
    google_graph_set_callback_script_html(),
    google_graph_corechart_script_html(),
    google_graph_function_script_open(),
    google_graph_datatable_statement() );

char *google_graph_document_close_html(
    google_graph_options_JSON(),
    google_graph_instantiation_statement(),
    google_graph_draw_statement(),
    google_graph_function_script_close(),
    google_graph_set_callback_script_html(),
    google_graph_division_tag() );

/* Attributes */
GOOGLE_FILENAME *google_filename;
LIST *google_graph_point_list;
char *document_open_html;
char *document_close_html;
```



GOOGLE_GRAPH (2)

```
/* Usage */
char *google_graph_visualization_function_name(
    int chart_number );

/* Process */
sprintf(
    static visualization_function_name[],
    "drawVisualization%d",
    chart_number );

/* Usage */
char *google_graph_id_string(
    int chart_number );

/* Process */
sprintf(
    static string[],
    "chart_id%d",
    chart_number );

/* Usage */
char *google_graph_loader_script_html(
    void );

/* Process */
return
"<script type=\"text/javascript\""
" src=\"https://www.gstatic.com/charts/loader.js\">"
"</script>";

/* Usage */
char *google_graph_corechart_script_html(
    void );

/* Process */
return
"<script type=\"text/javascript\">"
"google.charts.load('current',"
"{packages: ['corechart']});</script>";

/* Usage */
char *google_graph_function_script_open(
    char *google_graph_visualization_function_name() );

/* Process */
sprintf(
    static function_script_open[]
    "<script type=\"text/javascript\">\n"
    "function %s()\n"
    "{",
    google_graph_visualization_function_name );

/* Usage */
char *google_graph_datatable_statement(
    char *google_graph_number_attribute_label() );

/* Process */
sprintf(
    static datatable_statement[],
    "\tvar data = "
    "new google.visualization.arrayToDataTable([\n"
    "\t\t['','%s'],",
    google_graph_number_attribute_label );

/* Usage */
char *google_graph_options_JSON(
    char *chart_title );

/* Process */
sprintf(
    static options_JSON[],
    "\tvar options = {\n"
    "\t\tttitle: \"%s\"};",
    chart_title );

/* Usage */
char *google_graph_instantiation_statement(
    char *google_graph_id_string,
    char *google_graph_function
        /* either LineChart or ColumnChart */ );

/* Process */
sprintf(
    static instantiation_statement[],
    "var chart = new google.visualization.%s(\n"
    "\tdocument.getElementById('%s'));",
    google_graph_function,
    google_graph_id_string );

/* Usage */
char *google_graph_draw_statement(
    void );

/* Process */
return
"chart.draw(data,options);"

/* Usage */
char *google_graph_function_script_close(
    void );

/* Process */
return "}\n"</script>";

/* Usage */
char *google_graph_set_callback_script_html(
    char *google_graph_visualization_function_name() );

/* Process */
sprintf(
    static callback_script_html[],
    "<script type=\"text/javascript\">\n"
    "google.setOnloadCallback(%s);\n"
    "</script>",
    google_graph_visualization_function_name );

/* Usage */
char *google_graph_division_tag(
    char *google_graph_id_string() );

/* Process */
sprintf(
    static division_tag[],
    "<div id=\"%s\" style=\"\n"
    "\tposition: absolute;\n"
    "\tleft: 10px;\n"
    "\ttop: 10px;\n"
    "\twidth: 900px;\n"
    "\theight: 600px;\n"
    "\tborder-width: thin;\n"
    "\tborder-style: solid;\n"
    "\tborder-color: teal\"/>",
    google_graph_id_string );
```

GOOGLE_GRAPH (3)

```
/* Usage */
char *google_graph_document_open_html(
    char *google_graph_loader_script_html(),
    char *google_graph_set_callback_script_html(),
    char *google_graph_function_script_open(),
    char *google_graph_datatable_statement() );

/* Process */
char html[];
char *ptr = html;

char *document_type_html();

char *document_standard_string();

ptr +=
    sprintf(
        ptr,
        "%s\n",
        document_html(
            document_type_html(),
            document_standard_string() ) );

ptr +=
    sprintf(
        ptr,
        "%s\n",
        document_head_open_tag() );

ptr +=
    sprintf(
        ptr,
        "%s\n",
        google_graph_loader_script_html );

ptr +=
    sprintf(
        ptr,
        "%s\n",
        google_graph_corechart_script_html );

ptr +=
    sprintf(
        ptr,
        "%s\n",
        google_graph_script_function_open );

ptr +=
    sprintf(
        ptr,
        "%s",
        google_graph_datatable_statement );

/* Usage */
char *google_graph_document_close_html(
    char *google_graph_options_JSON(),
    char *google_graph_instantiation_statement(),
    char *google_graph_draw_statement(),
    char *google_graph_function_script_close(),
    char *google_graph_set_callback_script_html(),
    char *google_graph_division_tag() );

/* Process */
sprintf(
    document_close_html[],
    "%s\n"
    "%s\n"
    "%s\n"
    "%s\n"
    "%s\n"
    "</head>\n"
    "<body>\n"
    "%s\n"
    "</body>\n"
    "</html>",
    google_graph_options_JSON,
    google_graph_instantiation_statement,
    google_graph_draw_statement,
    google_graph_function_script_close,
    google_graph_set_callback_script_html,
    google_graph_division_tag );

/* Driver */
void google_graph_output(
    GOOGLE_GRAPH *google_graph,
    FILE *output_file );

/* Process */
fprintf(
    output_file,
    "%s\n",
    google_graph->document_open_html );

for GOOGLE_GRAPH_POINT *google_graph_point in
    google_graph->google_graph_point_list
{
    fprintf(
        output_file,
        "%s",
        google_graph_point->javascript );

    if ( list_last_boolean(
            google_graph->
                google_graph_point_list ) )
    {
        fprintf( output_file, "]);" );
    }
    else
    {
        fprintf( output_file, "," );
    }

    fprintf( output_file, "\n" );
}

fprintf(
    output_file,
    "%s\n",
    google_graph->document_close_html );
```

GOOGLE_WINDOW

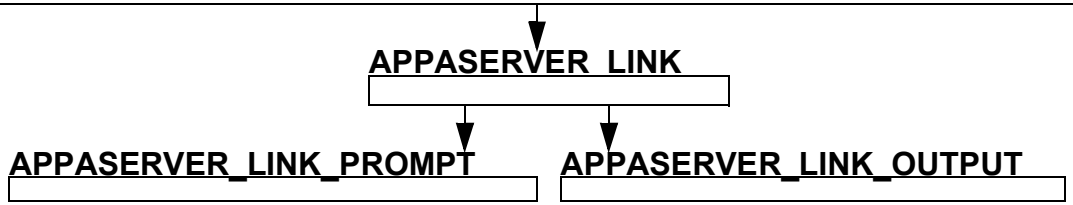
```
/* Usage */
GOOGLE_WINDOW *google_window_new(
    GOOGLE_FILENAME *google_filename,
    char *sub_title,
    pid_t process_id );

/* Process */
GOOGLE_WINDOW *google_window_calloc(
    void );

if ( google_filename )
{
    char *google_prompt_target(
        (char *)0 /* datatype_name */,
        process_id );

    char *onload_string =
        static char *chart_window_onload_string(
            google_filename->http_prompt_filename,
            google_prompt_target() );
}

char *document_body_onload_open_tag(
    JAVASCRIPT_WAIT_OVER,
    onload_string /* javascript_replace */,
    (char *)0 /* menu_onload_string */ );

char *date_time_now( date_utc_offset() );

char *google_window_screen_title(
    char *date_time_now() );

char *chart_window_sub_title_display(
    char *sub_title );

char *html =
    chart_window_html(
        document_body_onload_open_tag(),
        google_window_screen_title(),
        chart_window_sub_title_display() );

free( document_body_onload_open_tag() );

free( chart_window_sub_title_display() );

/* Attributes */
char *google_prompt_target;
char *onload_string;
char *date_time_now;
char *screen_title;
char *html;
```

GOOGLE_PROMPT

```
/* Usage */
GOOGLE_PROMPT *google_prompt_new(
    GOOGLE_FILENAME *google_filename,
    char *chart_filename_key(),
    pid_t process_id );

/* Process */
GOOGLE_PROMPT *google_prompt_calloc(
    void );

char *google_prompt_target(
    chart_filename_key,
    process_id );

char *google_prompt_html(
    google_filename->http_prompt_filename,
    chart_filename_key,
    google_prompt_target() );

/* Usage */
char *google_prompt_target(
    char *chart_filename_key,
    pid_t process_id );

/* Process */
if ( chart_filename_key )
{
    sprintf( static target[],
        "google_%s_%d",
        chart_filename_key,
        process_id );
}
else
{
    sprintf( static target[],
        "google_%d",
        process_id );
}

/* Usage */
char *google_prompt_html(
    char *http_prompt_filename,
    char *chart_filename_key(),
    char *google_prompt_target() );

/* Process */
char html[];
char *ptr = html;

ptr += sprintf( ptr,
    "<tr><td><a href=\"%s\" target=%s>View ",
    http_prompt_filename,
    google_prompt_target() );

if ( chart_filename_key )
{
    ptr += sprintf( ptr,
        "%s ",
        string_initial_capital(
            buffer[],
            filename_key ) );
}

sprintf( ptr, "chart.</a>" );

/* Attributes */
char *target;
char *html;
```

GOOGLE_FILENAME

```
/* Usage */
GOOGLE_FILENAME *google_filename_new(
    char *application_name,
    char *session_key,
    char *data_directory,
    char *number_attribute_name,
    char *chart_filename_key );

/* Process */
GOOGLE_FILENAME *grace_filename_calloc(
    void );

pid_t google_filename_process_id(
    char *session_key );

APPASERVER_LINK *appaserver_link =
    appaserver_link_new(
        application_http_prefix(
            application_name ),
        application_server_address(),
        ( application_prepend_http_protocol_yn(
            application_name ) == 'y' ),
        data_directory,
        GOOGLE_FILENAME_STEM,
        application_name,
        google_filename_process_id(),
        session_key,
        chart_filename_key /* begin_date_string */,
        (char *)0 /* end_date_string */,
        "html" /* extension */ );

char *google_filename_http_output_filename(
    APPASERVER_LINK *appaserver_link );

char *google_filename_http_prompt_filename(
    APPASERVER_LINK *appaserver_link );

/* If operation */
char *appaserver_link_anchor_html(
    http_prompt_filename,
    number_attribute_name /* target_window */,
    "Link to google file" /* prompt_message */ );

/* Attributes */
pid_t process_id;
APPASERVER_LINK *appaserver_link;
char *http_output_filename;
char *http_prompt_filename;
char *appaserver_link_anchor_html;
```



GOOGLE_GRAPH_POINT

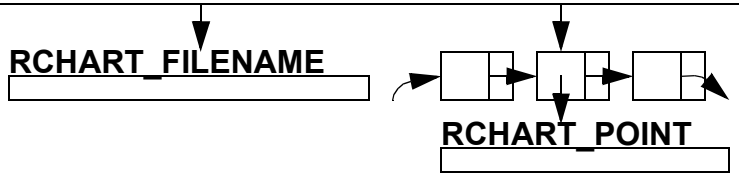
```
/* Usage */
LIST *google_graph_point_list(
    char *date_attribute_name,
    char *time_attribute_name,
    char *number_attribute_name,
    LIST *query_fetch_row_list );

/* Process */
LIST *point_list = list_new();

for QUERY_ROW *query_row in
    query_fetch_row_list
{
    QUERY_CELL *date_query_cell =
        query_cell_seek(
            date_attribute_name,
            query_row->cell_list );

    if ( query_chart->time_attribute_name )
    {
        QUERY_CELL *time_query_cell =
            query_cell_seek(
                time_attribute_name,
                query_row->cell_list );
    }

    QUERY_CELL *number_query_cell =
        query_cell_seek(
            number_attribute_name,
            query_row->cell_list );

    char *google_graph_point_javascript(
        date_query_cell,
        time_query_cell,
        number_query_cell );

    GOOGLE_GRAPH_POINT *google_graph_point =
        google_graph_point_new(
            google_graph_point_javascript() );

    list_set(
        point_list,
        google_graph_point_new() );
}

/* Usage */
char *google_graph_point_javascript(
    QUERY_CELL *date_query_cell,
    QUERY_CELL *time_query_cell,
    QUERY_CELL *number_query_cell );

/* Process */
char *google_graph_point_hhmm(
    QUERY_CELL *time_query_cell );

char *google_graph_point_date_javascript(
    date_query_cell->select_datum
        /* yyyy_mm_dd */,
    google_graph_point_hhmm() );

GOOGLE_GRAPH_VALUE *google_graph_value =
    google_graph_value_new(
        number_query_cell->select_datum );

sprintf(
    javascript[],
    "[new %s,%s]",
    google_graph_point_date_javascript(),
    google_graph_value->display );

/* Usage */
char *google_graph_point_date_javascript(
    char *yyyy_mm_dd,
    char *hhmm );

/* Process */

/* Usage */
GOOGLE_GRAPH_POINT *
google_graph_point_new(
    char *google_graph_point_javascript() );

/* Process */
GOOGLE_GRAPH_POINT *
google_graph_point_calloc(
    void );

/* Attributes */
char *javascript;
```

GOOGLE_GRAPH_VALUE

```
/* Usage */
GOOGLE_GRAPH_VALUE *google_graph_value_new(
    char *value_string );

/* Process */
GOOGLE_GRAPH_VALUE *
    google_graph_value_calloc(
        void );

boolean google_graph_value_null_boolean(
    char *value_string );

char *google_graph_value_display(
    value_string,
    google_graph_value_null_boolean() );

/* Usage */
char *google_graph_value_display(
    char *value_string,
    boolean google_graph_value_null_boolean() );

/* Process */
if ( google_graph_value_null_boolean )
    return “undefined”;
else
    return value_string;

/* Attributes */
boolean null_boolean;
char *display;
```

RCHART

```
/* Usage */
RCHART *rchart_new(
    char *application_name,
    char *session_key,
    char *data_directory,
    char *date_attribute_name,
    char *time_attribute_name,
    char *number_attribute_name,
    LIST *query_fetch_row_list,
    char *sub_title,
    char *chart_title(),
    char *chart_filename_key() );

/* Process */
RCHART *rchart_calloc(
    void );

RCHART_FILENAME *rchart_filename =
    rchart_filename_new(
        application_name,
        session_key,
        data_directory,
        chart_filename_key );

program char *rchart_file_header_string(
    void );

LIST *rchart_point_list(
    date_attribute_name,
    time_attribute_name,
    number_attribute_name,
    query_fetch_row_list );

if ( !list_length( rchart_point_list() ) return this;

char *rchart_point_date_combine_string =
    rchart_point_combine_string(
        date_attribute_name,
        rchart_point_list(),
        1 /* date_time_boolean */ );

char *rchart_point_value_combine_string =
    rchart_point_combine_string(
        number_attribute_name,
        rchart_point_list(),
        0 /* not date_time_boolean */ );

char *rchart_file_trailer_string(
    char *date_attribute_name,
    char *number_attribute_name,
    char *sub_title,
    char *chart_title,
    char *rchart_filename->pdf_output_filename );

void rchart_write(
    rchart_filename->R_output_filename,
    rchart_file_header_string(),
    rchart_point_date_combine_string,
    rchart_point_value_combine_string,
    rchart_file_trailer_string() );

char *rchart_script_system_string(
    rchart_filename->R_output_filename );

/* Usage */
char *rchart_script_system_string(
    char *rchart_filename->R_output_filename );

/* Process */
snprintf(
    system_string[],
    sizeof ( system_string ),
    "Rscript.sh %s /dev/null",
    R_output_filename );

/* Usage */
void rchart_free(
    RCHART *rchart );

/* Process */
free( rchart->rchart_point_date_combine_string );
free( rchart->rchart_point_value_combine_string );
free( rchart->file_trailer_string );
free( rchart->script_system_string );
rchart_point_list_free( rchart->rchart_point_list );
free( rchart );

/* Usage */
void rchart_write(
    char *R_output_filename,
    char *rchart_file_header_string,
    char *rchart_point_date_combine_string,
    char *rchart_point_value_combine_string,
    char *rchart_file_trailer_string );

/* Process */
FILE *appaserver_write_file(
    R_output_filename );

fprintf(
    appaserver_write_file(),
    "%s\n%s\n%s\n%s\n",
    rchart_file_header_string,
    rchart_point_date_combine_string,
    rchart_point_value_combine_string,
    rchart_file_trailer_string );

fclose( appaserver_write_file() );

/* Attributes */
RCHART_FILENAME *rchart_filename;
char *file_header_string;
LIST *rchart_point_list;
char *rchart_point_date_combine_string;
char *rchart_point_value_combine_string;
char *file_trailer_string;
char *script_system_string;
```

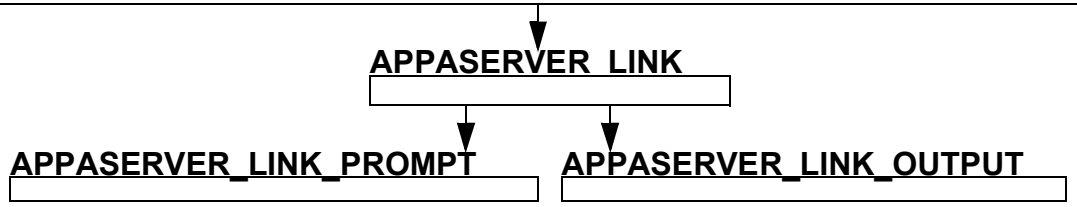


RCHART_PROMPT

```
/* Usage */
RCHART_PROMPT *rchart_prompt_new(
    RCHART_FILENAME *rchart_filename,
    char *filename_key,
    pid_t process_id );

/* Process */
RCHART_PROMPT *rchart_prompt_calloc(
    void );

char *rchart_prompt_pdf_target(
    filename_key,
    process_id );

static char *rchart_prompt_pdf_anchor_html(
    char *rchart_filename->pdf_prompt_filename,
    char *filename_key,
    char *rchart_prompt_pdf_target() );

char *rchart_prompt_R_target(
    filename_key,
    process_id );

static char *rchart_prompt_R_anchor_html(
    char *rchart_filename->R_prompt_filename,
    char *filename_key,
    char *rchart_prompt_R_target() );

char *rchart_prompt_html(
    rchart_prompt_pdf_anchor_html(),
    rchart_prompt_R_anchor_html() );

/* Usage */
char *rchart_prompt_pdf_target(
    char *filename_key,
    pid_t process_id );

/* Process */
if ( filename_key )
{
    snprintf(
        static target[],
        sizeof ( target ),
        "R_pdf_%s_%d",
        filename_key,
        process_id );
}
else
{
    snprintf(
        static target[],
        sizeof ( target ),
        "R_pdf_%d",
        process_id );
}

/* Usage */
char *rchart_prompt_R_target(
    char *filename_key,
    pid_t process_id );

/* Process */
if ( filename_key )
{
    snprintf(
        static target[],
        sizeof ( target ),
        "R_%s_%d",
        filename_key,
        process_id );
}
else
{
    snprintf(
        static target[],
        sizeof ( target ),
        "R_%d",
        process_id );
}

/* Usage */
char *rchart_prompt_html(
    char *rchart_prompt_pdf_anchor_html,
    char *rchart_prompt_R_anchor_html );

/* Process */
snprintf(
    html[],
    sizeof ( html ),
    "<tr><td>%s<td>%s",
    rchart_prompt_pdf_anchor_html,
    rchart_prompt_R_anchor_html );

/* Attributes */
char *pdf_target;
char *pdf_anchor_html;
char *R_target;
char *R_anchor_html;
char *html;
```

RCHART_FILENAME

```
/* Usage */
RCHART_FILENAME *rchart_filename_new(
    char *application_name,
    char *session_key,
    char *data_directory,
    char *chart_filename_key );

/* Process */
RCHART_FILENAME *rchart_filename_calloc(
    void );

pid_t rchart_filename_process_id(
     char *session_key );

APPASERVER_LINK *appaserver_link =
     appaserver_link_new(
          application_http_prefix(
               application_name ),
          application_server_address(),
          ( application_prepend_http_protocol_yn(
               application_name ) == 'y' ),
          appaserver_parameter_data_directory,
          RCHART_FILENAME_STEM,
          application_name,
          rchart_filename_process_id(),
          session_key,
          chart_filename_key /* begin_date_string */,
          (char *)0 /* end_date_string */,
          (char *)0 /* extension */ );

char *rchart_filename_R_output_filename(
     APPASERVER_LINK *appaserver_link );

char *rchart_filename_R_prompt_filename(
     APPASERVER_LINK *appaserver_link );

char *rchart_filename_pdf_output_filename(
     APPASERVER_LINK *appaserver_link );

char *rchart_filename_pdf_prompt_filename(
     APPASERVER_LINK *appaserver_link );

/* Attributes */
pid_t process_id;
APPASERVER_LINK *appaserver_link;
char *R_output_filename;
char *R_prompt_filename;
char *pdf_output_filename;
char *pdf_prompt_filename;
```



RCHART_WINDOW

```
/* Usage */
RCHART_WINDOW *rchart_window_new(
    RCHART_FILENAME *rchart_filename,
    char *sub_title,
    char *filename_key,
    pid_t process_id );

/* Process */
RCHART_WINDOW *rchart_window_calloc(
    void );

if ( rchart_filename )
{
    char *rchart_prompt_pdf_target(
        filename_key,
        process_id );

    char *onload_string =
        static char *chart_window_onload_string(
            rchart_filename->pdf_prompt_filename,
            rchart_prompt_pdf_target() );
}

char *document_body_onload_open_tag(
    JAVASCRIPT_WAIT_OVER,
    onload_string /* javascript_replace */,
    (char *)0 /* menu_onload_string */ );

char *date_time_now( date_utc_offset() );

char *rchart_window_screen_title(
    char *date_time_now() );

char *chart_window_sub_title_display(
    char *sub_title );

char *html =
    char *chart_window_html(
        document_body_onload_open_tag(),
        chart_window_screen_title(),
        chart_window_sub_title_display() );

free( document_body_onload_open_tag() );

free( chart_window_sub_title_display() );

/* Driver */
printf(   "%s\n", rchart_window->html );

printf( "%s\n", rchart_prompt->html );

/* Attributes */
char *onload_string;
char *date_time_now;
char *screen_title;
char *html;
```

RCHART_POINT

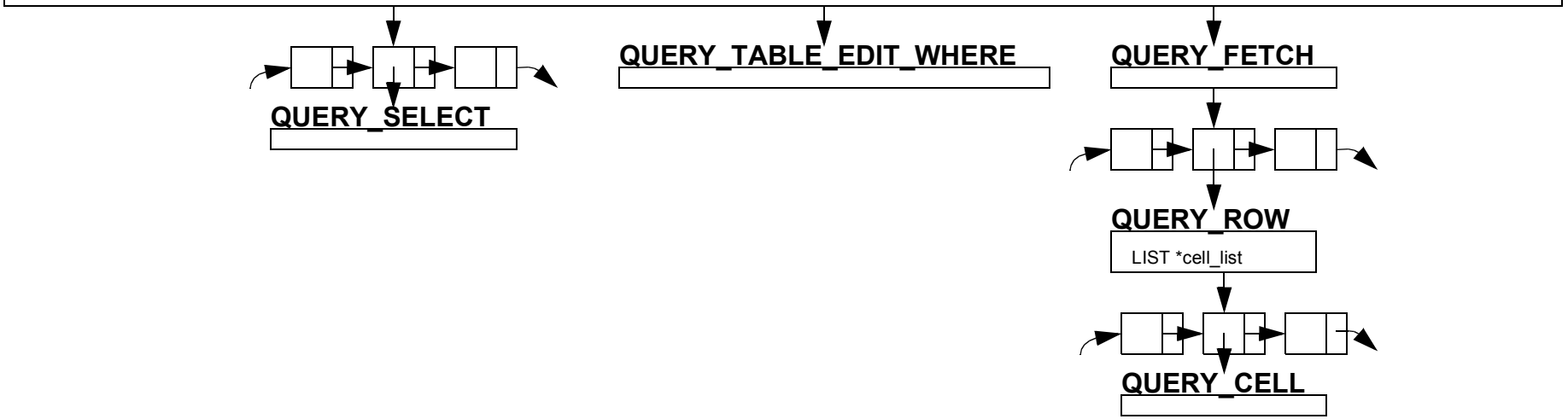
```
/* Usage */
LIST *rchart_point_list(
    char *date_attribute_name,
    char *time_attribute_name,
    char *number_attribute_name,
    LIST *query_fetch_row_list );

/* Process */
LIST *point_list = list_new();

for QUERY_ROW *query_row in
    query_fetch_row_list
{
    QUERY_CELL *number_query_cell =
        query_cell_seek(
            number_attribute_name,
            query_row->cell_list );

    boolean rchart_point_null_boolean(
        char *number_query_cell->select_datum );

    if ( rchart_point_null_boolean() ) continue;

    QUERY_CELL *date_query_cell =
        query_cell_seek(
            date_attribute_name,
            query_row->cell_list );

    if ( time_attribute_name )
    {
        QUERY_CELL *time_query_cell =
            query_cell_seek(
                time_attribute_name,
                query_row->cell_list );
    }

    char *rchart_point_date_time_string(
        date_query_cell,
        time_query_cell );

    double rchart_point_value(
        QUERY_CELL *number_query_cell );

    RCHART_POINT *rchart_point_new(
        rchart_point_date_time_string(),
        rchart_point_value() );

    list_set(
        point_list,
        rchart_point_new() );
}

/* Usage */
char *rchart_point_date_time_string(
    QUERY_CELL *date_query_cell,
    QUERY_CELL *time_query_cell );

/* Process */
char *rchart_point_hhmmss(
    QUERY_CELL *time_query_cell );

/* Usage */
RCHART_POINT *rchart_point_new(
    char *rchart_point_date_time_string,
    double rchart_point_value );

/* Process */
RCHART_POINT *rchart_point_calloc(
    void );

/* Usage */
char *rchart_point_combine_string(
    char *attribute_name,
    LIST *rchart_point_list,
    boolean date_time_boolean );

/* Process */
char combine_string[];
char *ptr = combine_string;
register int first_time = 1;

ptr += sprintf( ptr,
    "%s <- c(\n",
    attribute_name );

for RCHART_POINT *rchart_point in rchart_point_list
{
    if ( first_time )
        first_time = 0;
    else
    if ( list_still_more( rchart_point_list ) )
        ptr += sprintf( ptr, ",\n" );

    static char *rchart_point_string(
        char *rchart_point->date_time_string,
        double rchart_point->value,
        boolean date_time_boolean );

    ptr += sprintf( ptr,
        "%s",
        rchart_point_string() ;
}

ptr += sprintf( ptr, ")" );

/* Usage */
void rchart_point_list_free(
    LIST *rchart_point_list );

/* Process */
for RCHART_POINT *rchart_point in rchart_point_list
{
    free( rchart_point->date_time_string );
    free( rchart_point );
}

list_free_container( rchart_point_list );

/* Attributes */
char *date_time_string;
double value;
```

QUERY_CHART (1)

```
/* Usage */
QUERY_CHART *query_chart_new(
    char *application_name,
    char *login_name,
    char *folder_name,
    char *security_entity_where,
    DICTIONARY *dictionary_separate->
        query_dictionary,
    LIST *dictionary_separate->no_display_name_list,
    LIST *folder_attribute_append_isa_list,
    LIST *relation_mto1_non_isa_list,
    LIST *relation_mto1_isa_list );

/* Process */
QUERY_CHART *query_chart_calloc(
    void );

char *query_chart_date_attribute_name(
    LIST *folder_attribute_append_isa_list );

if ( !query_chart_date_attribute_name() ) return NULL;

char *query_chart_time_attribute_name(
    LIST *folder_attribute_append_isa_list );

LIST *query_chart_number_attribute_name_list(,
    LIST *no_display_name_list,
    LIST *folder_attribute_append_isa_list );

if ( !list_length(
        query_chart_number_attribute_name_list() ) )
            return (QUERY_CHART *)0;

LIST *query_chart_select_list(
    application_name,
    folder_name,
    list_length( relation_mto1_isa_list )
        /* relation_mto1_isa_list_length */,
    query_chart_date_attribute_name(),
    query_chart_time_attribute_name(),
    query_chart_number_attribute_name_list() );

char *query_select_list_string(
    query_chart_select_list(),
    (LIST *)0 /* common_name_list */ );

LIST *query_select_name_list(
    query_chart_select_list() );

char *query_from_string(
    folder_name,
    relation_mto1_isa_list,
    (char *)0 /* row_security_role_update_list_from */ );

QUERY_TABLE_EDIT_WHERE *
    query_table_edit_where =
        query_table_edit_where_new(
            application_name,
            folder_name,
            relation_mto1_non_isa_list,
            relation_mto1_isa_list,
            folder_attribute_append_isa_list,
            security_entity_where,
            query_dictionary,
            (char *)0 /* attribute_not_null */,
            (char *)0 /* join_where */ );

LIST *query_chart_order_name_list(
    char *query_chart_date_attribute_name(),
    char *query_chart_time_attribute_name() );

char *query_order_string(
    query_chart_order_name_list()
        /* folder_attribute_primary_key_list */,
    (LIST *)0 /* folder_attribute_name_list */,
    (DICTIONARY *)0 /* sort_dictionary */ );

char *query_system_string(
    application_name,
    query_select_list_string(),
    query_from_string(),
    query_table_edit_where->string,
    (char *)0 /* group_by_string */,
    query_order_string(),
    QUERY_CHART_MAX_ROWS );

QUERY_FETCH *query_fetch =
    query_fetch_new(
        folder_name,
        folder_attribute_append_isa_list,
        date_convert_international
            /* destination_enum */,
        query_select_name_list(),
        query_system_string(),
        0 /* not input_save_boolean */ );
```

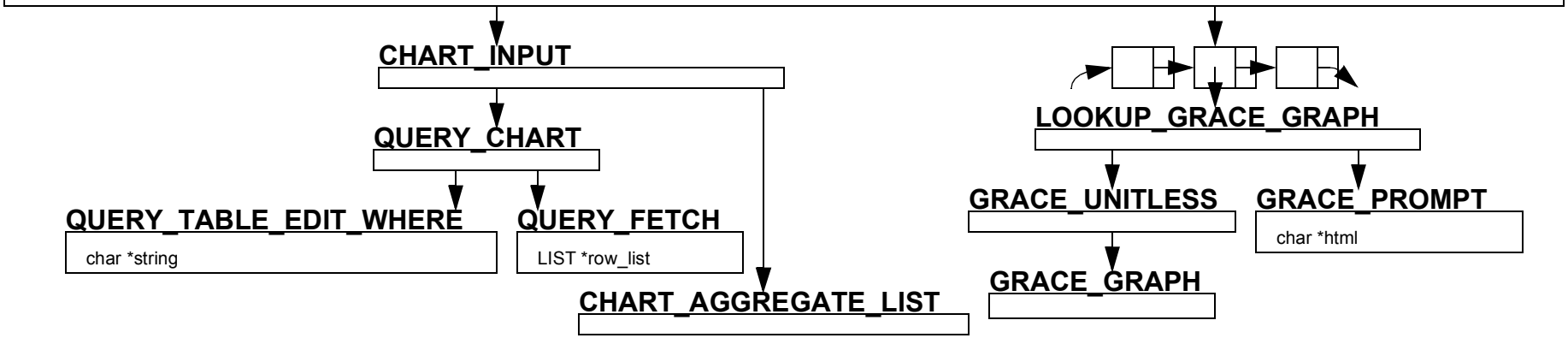
QUERY_CHART (2)

```
/* Usage */
LIST *query_chart_select_list(
    char *application_name,
    char *folder_name,
    int relation_mto1_isa_list_length,
    char *query_chart_date_attribute_name(),
    char *query_chart_time_attribute_name(),
    LIST *query_chart_number_attribute_name_list() );

/* Process */
LIST *select_list = list_new();

list_set(
    select_list,
    query_select_new(
        relation_mto1_isa_list_length,
        (char *)0 /* attribute_not_null */,
        folder_name,
        query_chart_date_attribute_name ) );

if ( query_chart_time_attribute_name )
{
    list_set(
        select_list,
        query_select_new(
            relation_mto1_isa_list_length,
            (char *)0 /* attribute_not_null */,
            folder_name,
            query_chart_time_attribute_name ) );
}

for char *number_attribute_name in
    query_chart_number_attribute_name_list
{
    list_set(
        select_list,
        query_select_new(
            0 /* relation_mto1_isa_list_length */,
            (char *)0 /* attribute_not_null */,
            (char *)0 /* folder_name */,
            number_attribute_name ) );
}

/* Usage */
void query_chart_free(
    QUERY_CHART *query_chart );

/* Process */
query_table_edit_where_free(
    query_chart->query_table_edit_where );

free( query_chart );

/* Attributes */
char *date_attribute_name;
char *time_attribute_name;
LIST *number_attribute_name_list;
LIST *select_list;
char *query_select_list_string;
LIST *query_select_name_list;
char *query_from_string;
QUERY_TABLE_EDIT_WHERE *
    query_table_edit_where;
LIST *order_name_list;
char *query_order_string;
char *query_system_string;
QUERY_FETCH *query_fetch;
```



LOOKUP_GRACE (1)

```
/* Usage */
LOOKUP_GRACE *lookup_grace_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *dictionary_string,
    char *appaserver_parameter->data_directory );

/* Process */
LOOKUP_GRACE *lookup_grace_calloc(
    void );

CHART_INPUT *chart_input =
    chart_input_new(
        application_name,
        login_name,
        role_name,
        folder_name,
        dictionary_string );

if ( !chart_input->row_list_length ) return NULL;

LIST *lookup_grace_graph_list = list_new();

for char *number_attribute_name
    chart_input->
        query_chart->
        number_attribute_name_list )
{
    char *chart_title(
        folder_name,
        number_attribute_name,
        aggregate_statistic_none
            /* enum aggregate_statistic */,
        aggregate_level_none
            /* enum aggregate_level */,
        (char *)0 /* where_string */ );

    char *lookup_grace_filename_key(
        number_attribute_name,
        aggregate_statistic_none
            /* enum aggregate_statistic */,
        aggregate_level_none
            /* enum aggregate_level */ );

    LOOKUP_GRACE_GRAPH *
        lookup_grace_graph_new(
            application_name,
            session_key,
            data_directory,
            chart_input->
                query_chart->
                query_table_edit_where->
                string /* sub_title */,
            chart_input->
                query_chart->
                date_attribute_name,
            chart_input->
                query_chart->
                time_attribute_name,
            chart_input->
                query_chart->
                query_fetch->
                row_list,
            chart_input->query_row_list_days_range
                /* horizontal_range_integer */,
            chart_title(),
            number_attribute_name
                /* grace_datatype_name */,
            lookup_grace_filename_key() );

    list_set(
        lookup_grace_graph_list,
        lookup_grace_graph_new() );
}
```



LOOKUP_GRACE (2)

```
/* Process (continued) */
if ( chart_input->chart_aggregate_list )
{
    LOOKUP_GRACE_AGGREGATE *
        lookup_grace_aggregate_new(
            application_name,
            session_key,
            folder_name,
            data_directory,
            chart_input->query_chart,
            chart_input->query_row_list_days_range,
            chart_input->
                chart_aggregate_list->
                chart_aggregate_attribute_list );

    list_set_list(
        lookup_grace_graph_list,
        lookup_grace_aggregate_new()->
            grace_graph_list );
}
```

```
GRACE_FILENAME *lookup_grace_filename(
    lookup_grace_graph_list );

GRACE_WINDOW *grace_window =
    grace_window_new(
        lookup_grace_filename(),
        chart_input->
            query_chart->
            query_table_edit_where->
            string /* sub_title */,
        (char *)0 /* filename_key */,
        getpid() /* process_id */ );

/* Usage */
GRACE_FILENAME *lookup_grace_filename(
    LIST *lookup_grace_graph_list );

/* Process */
if ( list_length( lookup_grace_graph_list ) == 1 )
{
    LOOKUP_GRACE_GRAPH *lookup_grace_graph =
        list_first(
            lookup_grace_graph_list );

    return
        lookup_grace_graph->
            grace_unitless->
            grace_setup->
            grace_filename;
}
else
{
    return NULL;
}

/* Attributes */
CHART_INPUT *chart_input;
LIST *lookup_grace_graph_list;
LOOKUP_GRACE_AGGREGATE *
    lookup_grace_aggregate;
GRACE_FILENAME *grace_filename;
GRACE_WINDOW *grace_window;
```

```
/* Driver */
void document_head_output_html_stream( stdout );

printf(
   "%s\n",
   lookup_grace->grace_window->html );

printf( "<table border=1>\n" );

printf( "<tr><th>PDF<th>GUI\n" );

for LOOKUP_GRACE_GRAPH *
   lookup_grace_graph
      in lookup_grace->lookup_grace_graph_list
{
   void grace_single_graph_pdf_create(
      lookup_grace_graph->
         grace_unitless->
            grace_setup->
            grace_ps_system_string,
      lookup_grace_graph->
         grace_unitless->
         grace_setup->
         grace_page,
      lookup_grace_graph->
         grace_unitless->
         grace_setup->
         grace_title,
      lookup_grace_graph->
         grace_unitless->
         grace_graph,
      lookup_grace_graph->
         grace_unitless->
         grace_setup->
         grace_pdf_system_string );

   printf(
      "%s\n",
      lookup_grace_graph->
         grace_prompt->
         html );
}

printf( "</table>\n" );

printf(
   "%s\n",
   lookup_grace->
      grace_window->
      grace_home_url_html );

document_close();
```

LOOKUP_GRACE_AGGREGATE (1)

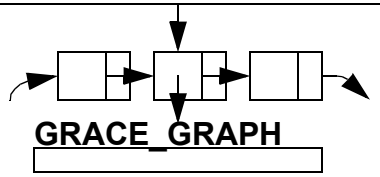
```
/* Usage */
LOOKUP_GRACE_AGGREGATE *
   lookup_grace_aggregate_new(
      char *application_name,
      char *session_key,
      char *folder_name,
      char *data_directory,
      QUERY_CHART *query_chart,
      int query_row_list_days_range,
      LIST *chart_aggregate_attribute_list );

/* Process */
LOOKUP_GRACE_AGGREGATE *
   lookup_grace_aggregate_calloc(
      void );

LIST *grace_graph_list = list_new();

for CHART_AGGREGATE_ATTRIBUTE *
   chart_aggregate_attribute in
      chart_aggregate_attribute_list
{
   if ( list_length(
         chart_aggregate_attribute->
            week_average_row_list ) )
   {
      char *chart_title(
         folder_name,
         chart_aggregate_attribute->
            number_attribute_name,
         average /* enum aggregate_statistic */,
         weekly /* enum aggregate_level */,
         (char *)0 /* where_string */ );

      char *chart_filename_key(\
         GRACE_FILENAME_STEM,
         chart_aggregate_attribute->
            number_attribute_name,
         average /* enum aggregate_statistic */,
         weekly /* enum aggregate_level */ );

LOOKUP_GRACE_GRAPH *
   lookup_grace_graph_new(
      application_name,
      session_key,
      data_directory,
      query_chart->
         query_table_edit_where->
         string /* sub_title */,
      query_chart->
         date_attribute_name,
      (char *)0 /* time_attribute_name */,
      chart_aggregate_attribute->
         week_average_row_list,
      query_row_list_days_range
         /* horizontal_range_integer */,
      chart_title(),
      chart_aggregate_attribute->
         number_attribute_name
         /* grace_datatype_name */,
      chart_filename_key() );

list_set(
   grace_graph_list,
   lookup_grace_graph_new() );

char *chart_title(
   folder_name,
   chart_aggregate_attribute->
      number_attribute_name,
      sum /* enum aggregate_statistic */,
      weekly /* enum aggregate_level */ );

char *chart_filename_key(
   GRACE_FILENAME_STEM,
   chart_aggregate_attribute->
      number_attribute_name,
   sum /* enum aggregate_statistic */,
   weekly /* enum aggregate_level */ );

LOOKUP_GRACE_GRAPH *
   lookup_grace_graph_new(
      application_name,
      session_key,
      data_directory,
      query_chart->
         query_table_edit_where->
         string /* sub_title */,
      query_chart->
         date_attribute_name,
      (char *)0 /* time_attribute_name */,
      chart_aggregate_attribute->
         week_sum_row_list,
      query_row_list_days_range
         /* horizontal_range_integer */,
      chart_title(),
      chart_aggregate_attribute->
         number_attribute_name
         /* grace_datatype_name */,
      chart_filename_key() );

   list_set(
      grace_graph_list,
      lookup_grace_graph_new() );
} /* if week_average_row_list */
```

LOOKUP_GRACE_AGGREGATE (2)

```
/* Process (continued) */
    if ( list_length(
            chart_aggregate_attribute->
               month_average_row_list ) )
    {
        char *chart_title(
            folder_name,
            chart_aggregate_attribute->
               number_attribute_name,
            average /* enum aggregate_statistic */,
            monthly /* enum aggregate_level */ );

        char *chart_filename_key(
            GRACE_FILENAME_STEM,
            chart_aggregate_attribute->
               number_attribute_name,
            average /* enum aggregate_statistic */,
            monthly /* enum aggregate_level */ );

        LOOKUP_GRACE_GRAPH *
            lookup_grace_graph_new(
               application_name,
               session_key,
               data_directory,
               query_chart->
                  query_table_edit_where->
                  string /* sub_title */,
               query_chart->
                  date_attribute_name,
               (char *)0 /* time_attribute_name */,
               chart_aggregate_attribute->
                  month_average_row_list,
               query_row_list_days_range
                  /* horizontal_range_integer */,
               chart_title(),
               chart_aggregate_attribute->
                  number_attribute_name
                  /* grace_datatype_name */,
               chart_filename_key() );

        list_set(
            grace_graph_list,
            lookup_grace_graph_new() );

        char *chart_title(
            folder_name,
            chart_aggregate_attribute->
               number_attribute_name,
            sum /* enum aggregate_statistic */,
            monthly /* enum aggregate_level */ );

        char *chart_filename_key(
            GRACE_FILENAME_STEM,
            chart_aggregate_attribute->
               number_attribute_name,
            sum /* enum aggregate_statistic */,
            monthly /* enum aggregate_level */ );

        LOOKUP_GRACE_GRAPH *
            lookup_grace_graph_new(
               application_name,
               session_key,
               data_directory,
               query_chart->
                  query_table_edit_where->
                  string /* sub_title */,
               query_chart->
                  date_attribute_name,
               (char *)0 /* time_attribute_name */,
               chart_aggregate_attribute->
                  month_sum_row_list,
               query_row_list_days_range
                  /* horizontal_range_integer */,
               chart_title(),
               chart_aggregate_attribute->
                  number_attribute_name
                  /* grace_datatype_name */,
               chart_filename_key() );

        list_set(
            grace_graph_list,
            lookup_grace_graph_new() );
    } /* if month_average_row_list */
```

LOOKUP_GRACE_AGGREGATE (3)

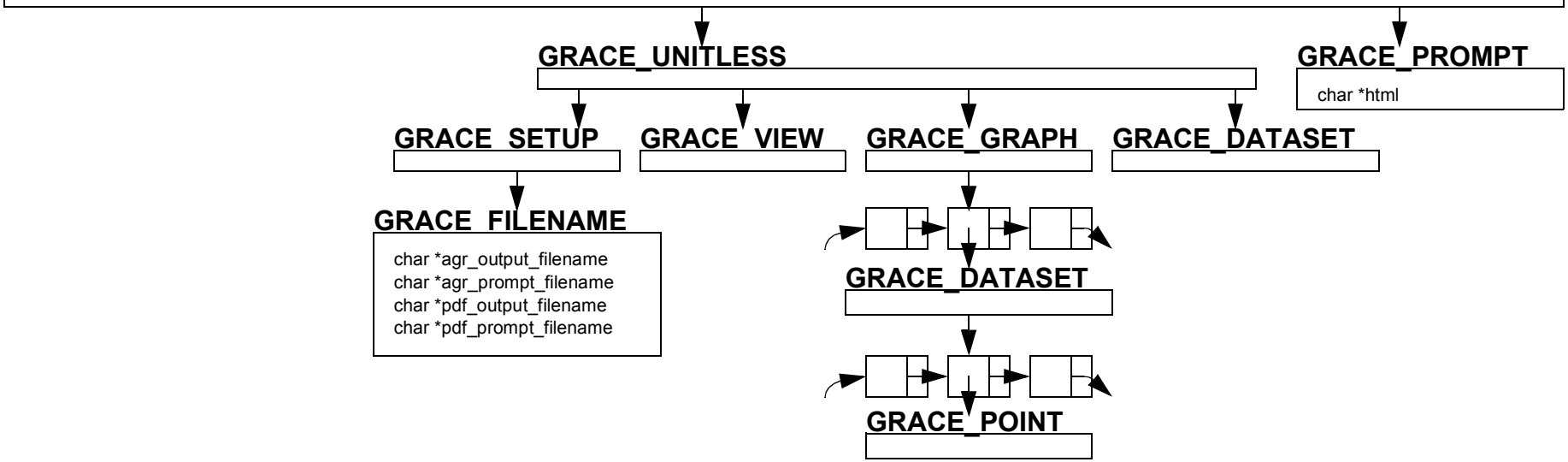
```
/* Process (continued) */
    if ( list_length(
            chart_aggregate_attribute->
               annual_average_row_list ) )
    {
        char *chart_title(
            folder_name,
            chart_aggregate_attribute->
               number_attribute_name,
            average /* enum aggregate_statistic */,
            annually /* enum aggregate_level */ );

        char *chart_filename_key(
            chart_aggregate_attribute->
               number_attribute_name,
            average /* enum aggregate_statistic */,
            annually /* enum aggregate_level */ );

        LOOKUP_GRACE_GRAPH *
            lookup_grace_graph_new(
               application_name,
               session_key,
               data_directory,
               query_chart->
                  query_table_edit_where->
                  string /* sub_title */,
               query_chart->
                  date_attribute_name,
               (char *)0 /* time_attribute_name */,
               chart_aggregate_attribute->
                  annual_average_row_list,
               query_row_list_days_range
                  /* horizontal_range_integer */,
               chart_title(),
               chart_aggregate_attribute->
                  number_attribute_name
                  /* grace_datatype_name */,
               chart_filename_key() );

        list_set(
            grace_graph_list,
            lookup_grace_graph_new() );

        char *chart_title(
            folder_name,
            chart_aggregate_attribute->
               number_attribute_name,
            sum /* enum aggregate_statistic */,
            annually /* enum aggregate_level */ );

        char *chart_filename_key(
            chart_aggregate_attribute->
               number_attribute_name,
            sum /* enum aggregate_statistic */,
            annually /* enum aggregate_level */ );

        LOOKUP_GRACE_GRAPH *
            lookup_grace_graph_new(
               application_name,
               session_key,
               data_directory,
               query_chart->
                  query_table_edit_where->
                  string /* sub_title */,
               query_chart->
                  date_attribute_name,
               (char *)0 /* time_attribute_name */,
               chart_aggregate_attribute->
                  annual_sum_row_list,
               query_row_list_days_range
                  /* horizontal_range_integer */,
               chart_title(),
               chart_aggregate_attribute->
                  number_attribute_name
                  /* grace_datatype_name */,
               chart_filename_key() );

            list_set(
               grace_graph_list,
               lookup_grace_graph_new() );
        } /* if annual_average_row_list */
    } /* for each chart_aggregate_attribute_list */

    /* Attribute */
    LIST *grace_graph_list;
```



LOOKUP_GRACE_GRAPH

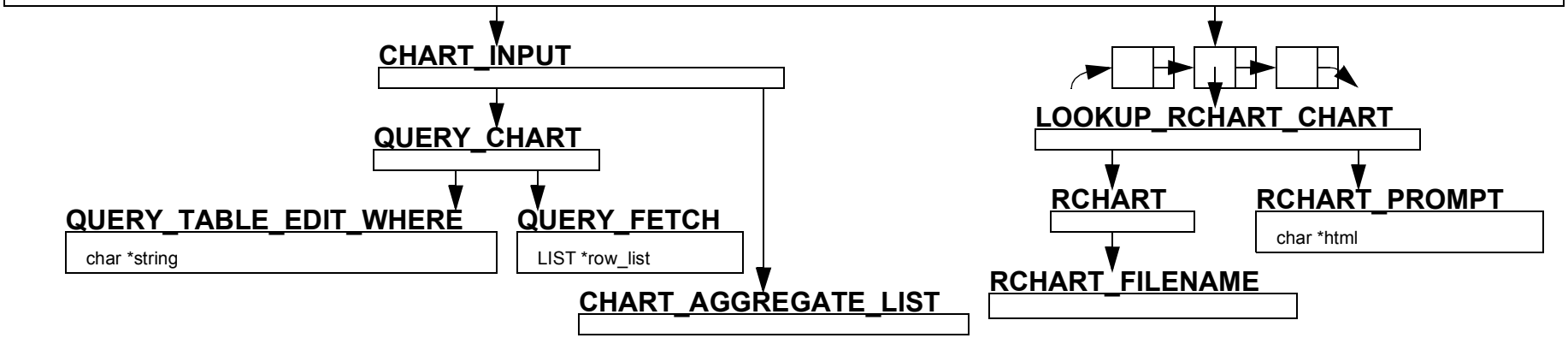
```
/* Usage */
LOOKUP_GRACE_GRAPH *
    lookup_grace_graph_new(
        char *application_name,
        char *session_key,
        char *data_directory,
        char *sub_title,
        char *date_attribute_name,
        char *time_attribute_name,
        LIST *query_fetch_row_list,
        int horizontal_range_integer,
        char *chart_title(),
        char *grace_datatype_name,
        char *chart_filename_key() );
```

```
/* Process */
LOOKUP_GRACE_GRAPH *
    lookup_grace_graph_calloc(
        void );

GRACE_UNITLESS *grace_unitless =
    grace_unitless_new(
        application_name,
        session_key,
        data_directory,
        sub_title,
        date_attribute_name,
        time_attribute_name,
        query_fetch_row_list,
        horizontal_range_integer,
        chart_title,
        chart_filename_key );

GRACE_PROMPT *grace_prompt =
    grace_prompt_new(
        grace_unitless->grace_setup->grace_filename,
        chart_filename_key,
        getpid() /* process_id */ );

/* Attributes */
GRACE_UNITLESS *grace_unitless;
GRACE_PROMPT *grace_prompt;
```



LOOKUP_RCHART (1)

```
/* Usage */
LOOKUP_RCHART *lookup_rchart_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *dictionary_string,
    char *appaserver_parameter->data_directory );

/* Process */
LOOKUP_RCHART *lookup_rchart_calloc(
    void );

CHART_INPUT *chart_input =
    chart_input_new(
        application_name,
        login_name,
        role_name,
        folder_name,
        dictionary_string );

if ( !chart_input->row_list_length ) return NULL;

LIST *lookup_rchart_chart_list = list_new();

for char *number_attribute_name
    chart_input->
        query_chart->
        number_attribute_name_list )
{
    char *chart_title(
        folder_name,
        number_attribute_name,
        aggregate_statistic_none
            /* enum aggregate_statistic */,
        aggregate_level_none
            /* enum aggregate_level */,
        (char *)0 /* where_string */ );

    char *chart_filename_key(
        RCHART_FILENAME_STEM,
        number_attribute_name,
        aggregate_statistic_none
            /* enum aggregate_statistic */,
        aggregate_level_none
            /* enum aggregate_level */ );

    LOOKUP_RCHART_CHART *
        lookup_rchart_chart_new(
            application_name,
            session_key,
            data_directory,
            chart_input->
                query_chart->
                query_table_edit_where->
                string /* sub_title */,
            chart_input->
                query_chart->
                date_attribute_name,
            chart_input->
                query_chart->
                time_attribute_name,
            number_attribute_name,
            chart_input->
                query_chart->
                query_fetch->
                row_list,
            chart_title(),
            chart_filename_key() );

    list_set(
        lookup_rchart_chart_list,
        lookup_rchart_chart_new() );
}
```

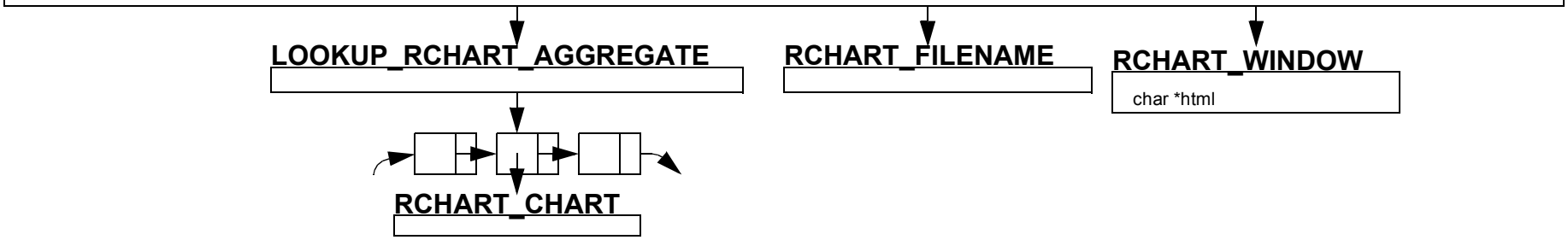


LOOKUP_RCHART (2)

```
/* Process (continued) */
if ( chart_input->chart_aggregate_list )
{
    LOOKUP_RCHART_AGGREGATE *
        lookup_rchart_aggregate_new(
            application_name,
            session_key,
            folder_name,
            data_directory,
            chart_input->query_chart,
            chart_input->
                chart_aggregate_list->
                chart_aggregate_attribute_list );

    list_set_list(
        lookup_rchart_chart_list,
        lookup_rchart_aggregate_new()->
            rchart_chart_list );
}

RCHART_FILENAME *lookup_rchart_filename(
    lookup_rchart_chart_list );

RCHART_WINDOW *rchart_window_new(
    lookup_rchart_filename(),
    chart_input->
        query_chart->
        query_table_edit_where->
        string /* sub_title */,
    (char *)0 /* filename_key */,
    getpid() /* process_id */ );
```

```
/* Usage */
RCHART_FILENAME *lookup_rchart_filename(
    LIST *lookup_rchart_chart_list );

/* Process */
if ( list_length( lookup_rchart_chart_list ) == 1 )
{
    LOOKUP_RCHART_CHART *lookup_rchart_chart =
        list_first(
            lookup_rchart_chart_list );

    return
        lookup_rchart_chart->
            rchart->
            rchart_filename;
}
else
{
    return NULL;
}

/* Driver */
void document_head_output_html_stream( stdout );

printf(
    "%s\n",
    lookup_rchart->rchart_window->html );

printf( "<table border=1>\n" );
```

```
printf( "<tr><th>PDF<th>GUI\n" );

for LOOKUP_RCHART_CHART *
    lookup_rchart_chart in
        lookup_rchart->lookup_rchart_chart_list
{
    system(
        lookup_rchart_chart->
            rchart->
            script_system_string );

    rchart_free( lookup_rchart_chart->rchart );

    printf(
        "%s\n",
        lookup_rchart_chart->
            rchart_prompt->
            html );
}

printf( "</table>\n" );

document_close();

/* Attributes */
CHART_INPUT *chart_input;
LIST *lookup_rchart_chart_list;
LOOKUP_RCHART_AGGREGATE *
    lookup_rchart_aggregate;
RCHART_FILENAME *rchart_filename;
RCHART_WINDOW *rchart_window;
```



LOOKUP_RCHART_AGGREGATE (1)

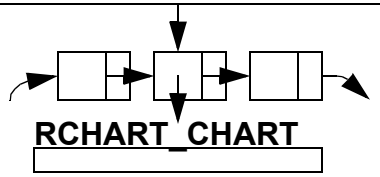
```
/* Usage */
LOOKUP_RCHART_AGGREGATE *
    lookup_rchart_aggregate_new(
        char *application_name,
        char *session_key,
        char *folder_name,
        char *data_directory,
        QUERY_CHART *query_chart,
        LIST *chart_aggregate_attribute_list );

/* Process */
LOOKUP_RCHART_AGGREGATE *
    lookup_rchart_aggregate_calloc(
        void );

LIST *rchart_chart_list = list_new();

for CHART_AGGREGATE_ATTRIBUTE *
    chart_aggregate_attribute in
        chart_aggregate_attribute_list
{
    if ( list_length(
            chart_aggregate_attribute->
                week_average_row_list ) )
    {
        char *chart_title(
            folder_name,
            chart_aggregate_attribute->
                number_attribute_name,
            average /* enum aggregate_statistic */,
            weekly /* enum aggregate_level */,
            (char *)0 /* where_string */ );

        char *chart_filename_key(\
            RCHART_FILENAME_STEM,
            chart_aggregate_attribute->
                number_attribute_name,
            average /* enum aggregate_statistic */,
            weekly /* enum aggregate_level */ );

        LOOKUP_RCHART_CHART *
            lookup_rchart_chart_new(
                application_name,
                session_key,
                data_directory,
                query_chart->
                    query_table_edit_where->
                    string /* sub_title */,
                query_chart->
                    date_attribute_name,
                (char *)0 /* time_attribute_name */,
                chart_aggregate_attribute->
                    week_average_row_list,
                chart_title(),
                chart_filename_key() );

        list_set(
            rchart_chart_list,
            lookup_rchart_chart_new() );

        char *chart_title(
            folder_name,
            chart_aggregate_attribute->
                number_attribute_name,
            sum /* enum aggregate_statistic */,
            weekly /* enum aggregate_level */ );

        char *chart_filename_key(
            RCHART_FILENAME_STEM,
            chart_aggregate_attribute->
                number_attribute_name,
            sum /* enum aggregate_statistic */,
            weekly /* enum aggregate_level */ );

        LOOKUP_RCHART_CHART *
            lookup_rchart_chart_new(
                application_name,
                session_key,
                data_directory,
                query_chart->
                    query_table_edit_where->
                    string /* sub_title */,
                query_chart->
                    date_attribute_name,
                (char *)0 /* time_attribute_name */,
                chart_aggregate_attribute->
                    week_sum_row_list,
                chart_title(),
                chart_filename_key() );

        list_set(
            rchart_chart_list,
            lookup_rchart_chart_new() );
    } /* if week_average_row_list */
```

LOOKUP_RCHART_AGGREGATE (2)

```
/* Process (continued) */
    if ( list_length(
            chart_aggregate_attribute->
               month_average_row_list ) )
    {
        char *chart_title(
            folder_name,
            chart_aggregate_attribute->
               number_attribute_name,
            average /* enum aggregate_statistic */,
            monthly /* enum aggregate_level */ );

        char *chart_filename_key(
            RCHART_FILENAME_STEM,
            chart_aggregate_attribute->
               number_attribute_name,
            average /* enum aggregate_statistic */,
            monthly /* enum aggregate_level */ );

        LOOKUP_RCHART_CHART *
            lookup_rchart_chart_new(
               application_name,
               session_key,
               data_directory,
               query_chart->
                  query_table_edit_where->
                  string /* sub_title */,
               query_chart->
                  date_attribute_name,
               (char *)0 /* time_attribute_name */,
               chart_aggregate_attribute->
                  month_average_row_list,
               chart_title(),
               chart_filename_key() );

        list_set(
            rchart_chart_list,
            lookup_rchart_chart_new() );

        char *chart_title(
            folder_name,
            chart_aggregate_attribute->
               number_attribute_name,
            sum /* enum aggregate_statistic */,
            monthly /* enum aggregate_level */ );

        char *chart_filename_key(
            RCHART_FILENAME_STEM,
            chart_aggregate_attribute->
               number_attribute_name,
            sum /* enum aggregate_statistic */,
            monthly /* enum aggregate_level */ );

        LOOKUP_RCHART_CHART *
            lookup_rchart_chart_new(
               application_name,
               session_key,
               data_directory,
               query_chart->
                  query_table_edit_where->
                  string /* sub_title */,
               query_chart->
                  date_attribute_name,
               (char *)0 /* time_attribute_name */,
               chart_aggregate_attribute->
                  month_sum_row_list,
               chart_title(),
               chart_filename_key() );

        list_set(
            rchart_chart_list,
            lookup_rchart_chart_new() );
    } /* if month_average_row_list */
```

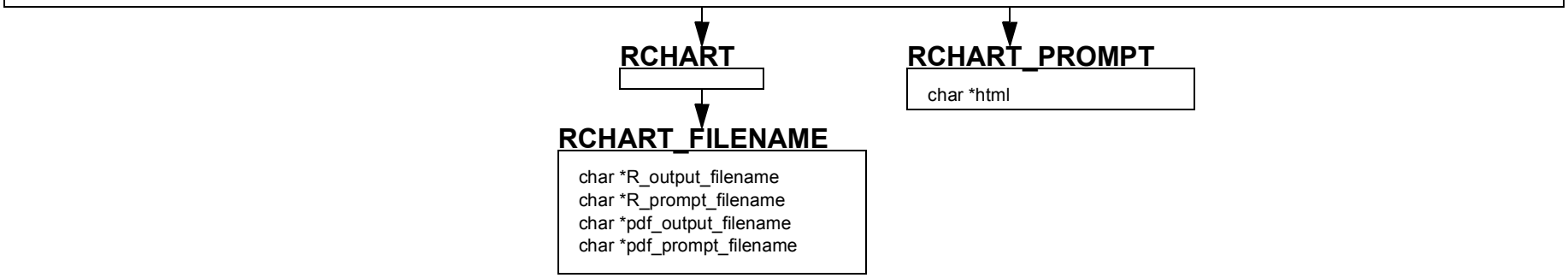
LOOKUP_RCHART_AGGREGATE (3)

```
/* Process (continued) */
    if ( list_length(
            chart_aggregate_attribute->
               annual_average_row_list ) )
    {
        char *chart_title(
            folder_name,
            chart_aggregate_attribute->
               number_attribute_name,
            average /* enum aggregate_statistic */,
            annually /* enum aggregate_level */ );

        char *chart_filename_key(
            chart_aggregate_attribute->
               number_attribute_name,
            average /* enum aggregate_statistic */,
            annually /* enum aggregate_level */ );

        LOOKUP_RCHART_CHART *
            lookup_rchart_chart_new(
               application_name,
               session_key,
               data_directory,
               query_chart->
                  query_table_edit_where->
                     string /* sub_title */,
               query_chart->
                  date_attribute_name,
               (char *)0 /* time_attribute_name */,
               chart_aggregate_attribute->
                  annual_average_row_list,
               chart_title(),
               chart_filename_key() );

        list_set(
            rchart_chart_list,
            lookup_rchart_chart_new() );

        char *chart_title(
            folder_name,
            chart_aggregate_attribute->
               number_attribute_name,
            sum /* enum aggregate_statistic */,
            annually /* enum aggregate_level */ );

        char *chart_filename_key(
            chart_aggregate_attribute->
               number_attribute_name,
            sum /* enum aggregate_statistic */,
            annually /* enum aggregate_level */ );

        LOOKUP_RCHART_CHART *
            lookup_rchart_chart_new(
               application_name,
               session_key,
               data_directory,
               query_chart->
                  query_table_edit_where->
                     string /* sub_title */,
               query_chart->
                  date_attribute_name,
               (char *)0 /* time_attribute_name */,
               chart_aggregate_attribute->
                  annual_sum_row_list,
               chart_title(),
               lookup_grace_filename_key() );

            list_set(
               rchart_chart_list,
               lookup_rchart_chart_new() );
        } /* if annual_average_row_list */
    } /* for each chart_aggregate_attribute_list */

    /* Attribute */
    LIST *rchart_chart_list;
```



LOOKUP_RCHART_CHART

```
/* Usage */
LOOKUP_RCHART_CHART *lookup_rchart_chart_new(
    char *application_name,
    char *session_key,
    char *data_directory,
    char *sub_title,
    char *date_attribute_name,
    char *time_attribute_name,
    char *number_attribute_name,
    LIST *query_fetch_row_list,
    char *chart_title(),
    char *chart_filename_key );

/* Process */
LOOKUP_RCHART_CHART *
    lookup_rchart_chart_calloc(
        void );

char *query_where_display(
    sub_title,
    0 /* max_length */);

RCHART *rchart_new(
    application_name,
    session_key,
    data_directory,
    date_attribute_name,
    time_attribute_name,
    number_attribute_name,
    query_fetch_row_list,
        query_where_display()
            /* sub_title */,
        chart_title,
        chart_filename_key );

RCHART_PROMPT *rchart_prompt_new(
    rchart->rchart_filename,
    chart_filename_key,
    getpid() /* process_id */ );

/* Attributes */
char *query_where_display;
RCHART *rchart;
RCHART_PROMPT *rchart_prompt;
```

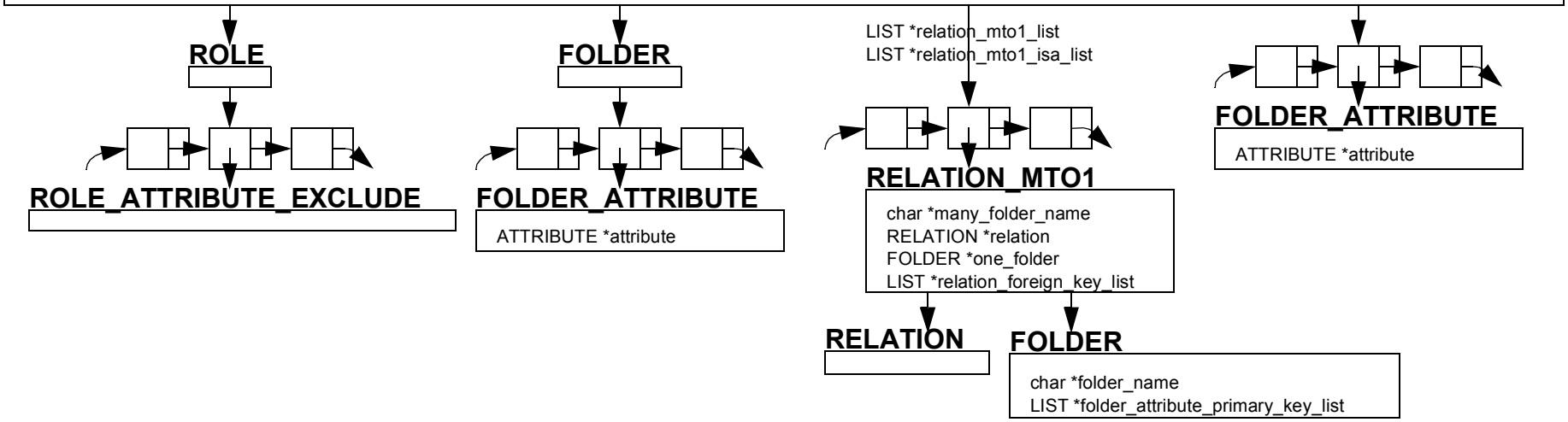


CHART

```
/* Usage */
char *chart_title(
    char *folder_name,
    char *number_attribute_name,
    enum aggregate statistic aggregate_statistic,
    enum aggregate_level aggregate_level,
    char *where_string );

/* Process */

/* Usage */
char *chart_filename_key(
    char *filename_stem,
    char *number_attribute_name,
    enum aggregate_statistic aggregate_statistic,
    enum aggregate_level aggregate_level );

/* Process */
if ( aggregate_statistic == aggregate_statistic_none )
{
    sprintf(
        static filename_key[],
        "%s_%s",
        filename_stem,
        number_attribute_name );
}
else
{
    sprintf(
        static filename_key[],
        "%s_%s_%s_%s",
        filename_stem,
        number_attribute_name,
        aggregate_statistic_string( aggregate_statistic ),
        aggregate_level_string( aggregate_level ) );
}

/* Usage */
char *chart_window_sub_title_display(
    char *sub_title );

/* Process */

/* Usage */
char *chart_window_html(
    char *document_body_onload_open_tag,
    char *screen_title,
    char *sub_title_display );

/* Process */
snprintf(
    html[],
    sizeof ( html ),
    "%s\n%s\n%s",
    document_body_onload_open_tag,
    screen_title,
    sub_title_display );

/* Usage */
static char *chart_window_onload_string(
    char *prompt_filename,
    char *target );

/* Process */
```

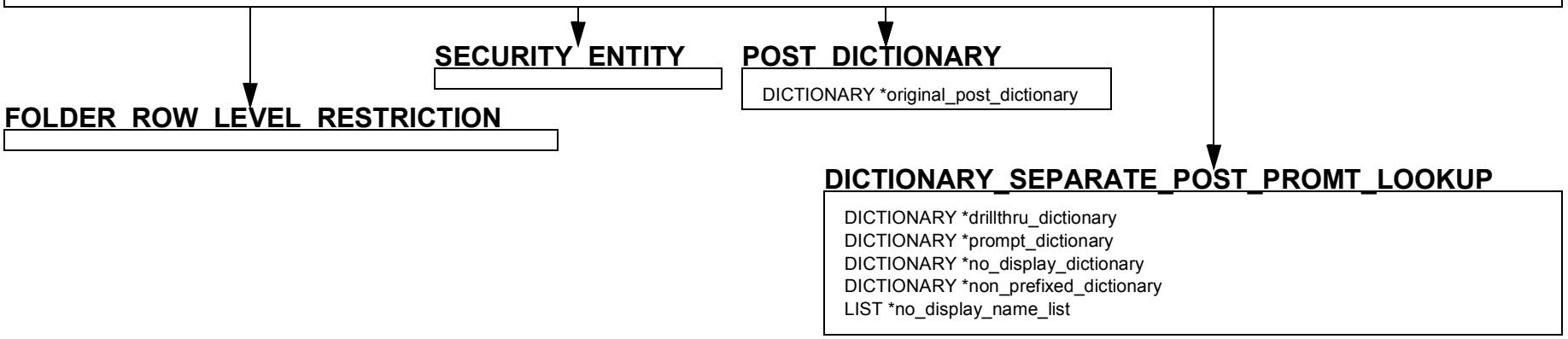
CHART_INPUT (1)

```
/* Usage */
CHART_INPUT *chart_input_new(
    char *application_name,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *dictionary_string );

/* Process */
CHART_INPUT *chart_input_calloc( void );

ROLE *role =
    role_fetch(
        role_name,
        1 /* fetch_role_attribute_exclude_list */ );

LIST *role_attribute_exclude_lookup_name_list(
    APPASERVER_LOOKUP_STATE,
    APPASERVER_UPDATE_STATE,
    role->role_attribute_exclude_list );

FOLDER *folder =
    folder_fetch(
        folder_name,
        /* Set to cache all folders for a role */
        role_name,
        role_attribute_exclude_lookup_name_list(),
        /* Sets folder_attribute_primary_list */
        /* Sets folder_attribute_primary_key_list */
        /* Sets folder_attribute_name_list */
        1 /* fetch_folder_attribute_list */,
        1 /* fetch_attribute */,
        1 /* cache_boolean */ );

LIST *relation_mto1_list(
    folder_name
        /* many_folder_name */,
    folder->folder_attribute_primary_key_list
        /* many_folder_primary_key_list */ );

LIST *relation_mto1_isa_list(
    (LIST *)0 /* mto1_isa_list Pass in null */,
    folder_name
        /* many_folder_name */,
    folder->folder_attribute_primary_key_list
        /* many_folder_primary_key_list */,
    0 /* not fetch_relation_one2m_list */,
    1 /* fetch_relation_mto1_list */ );

LIST *folder_attribute_append_isa_list(
    folder->folder_attribute_list,
    relation_mto1_isa_list() );

LIST *folder_attribute_date_name_list(
    folder_attribute_append_isa_list() );

LIST *folder_attribute_name_list(
    (char *)0 /* folder_name */,
    folder_attribute_append_isa_list() );
```



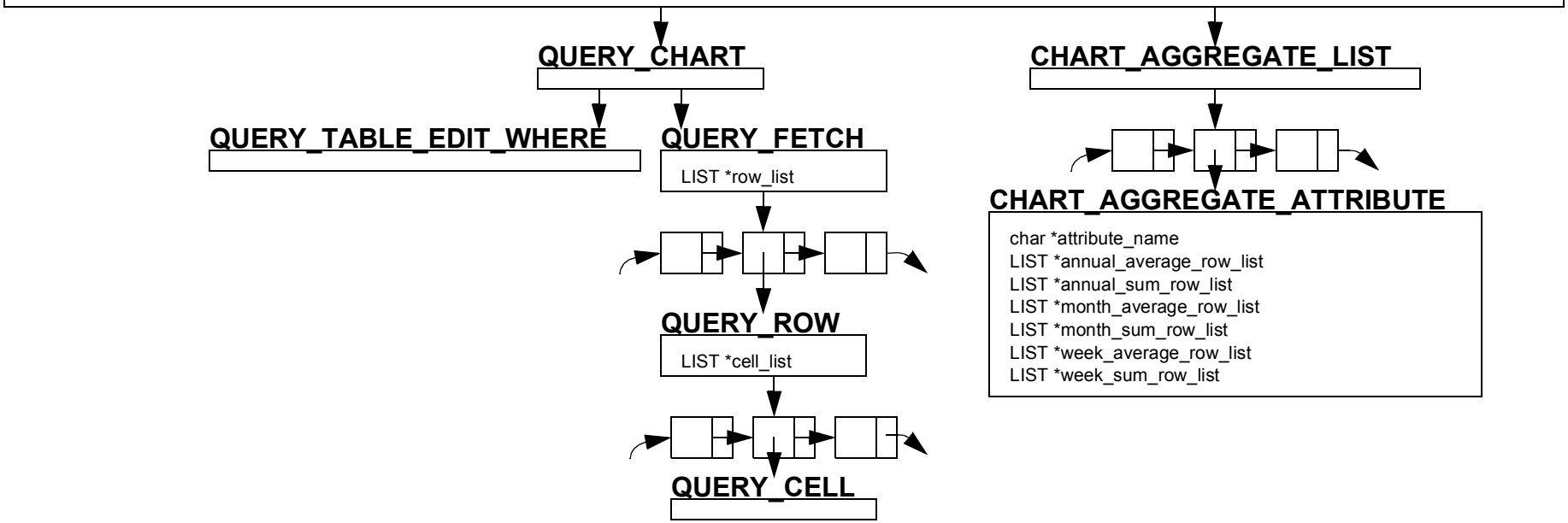
CHART_INPUT (2)

```
/* Process (continued) */
FOLDER_ROW_LEVEL_RESTRICTION *
    folder_row_level_restriction =
        folder_row_level_restriction_fetch(
            folder_name );

SECURITY_ENTITY *security_entity =
    security_entity_new(
        login_name,
        folder_name,
        folder_row_level_restriction->
            non_owner_forbid,
        role->override_row_restrictions );

POST_DICTIONARY *post_dictionary =
    post_dictionary_string_new(
        dictionary_string );

DICTIONARY_SEPARATE_POST_PROMPT_LOOKUP *
    dictionary_separate =
        dictionary_separate_post_prompt_lookup_new(
            post_dictionary->original_post_dictionary,
            application_name,
            login_name,
            relation_mto1_list(),
            folder_attribute_name_list(),
            folder_attribute_date_name_list(),
            folder_attribute_append_isa_list() );
```



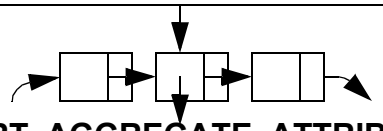
CHART_INPUT (3)

```
/* Process (continued) */
QUERY_CHART *query_chart =
    query_chart_new(
        application_name,
        login_name,
        folder_name,
        security_entity_where,
        dictionary_separate->
            non_prefixed_dictionary
                /* query_dictionary */,
        dictionary_separate->
            no_display_name_list,
        folder_attribute_append_isa_list,
        relation_mto1_list,
        relation_mto1_isa_list );

int chart_input_row_list_length(
    LIST *query_chart->
        query_fetch->
        row_list );

if ( !chart_input_row_list_length() ) return this;

int query_row_list_days_range(
    query_chart->
        date_attribute_name,
    query_chart->
        query_fetch->
        row_list );

if ( query_row_list_days_range() >= 180 )
{
    CHART_AGGREGATE_LIST *chart_aggregate_list =
        chart_aggregate_list_new(
            query_chart->date_attribute_name,
            query_chart->time_attribute_name,
            query_chart->query_select_name_list,
            query_chart->
                query_fetch->
                row_list,
            query_row_list_days_range() );
}

/* Attributes */
ROLE *role;
LIST *role_attribute_exclude_lookup_name_list;
FOLDER *folder;
LIST *relation_mto1_list;
LIST *relation_mto1_isa_list;
LIST *folder_attribute_append_isa_list;
LIST *folder_attribute_date_name_list;
LIST *folder_attribute_name_list;
FOLDER_ROW_LEVEL_RESTRICTION *
    folder_row_level_restriction;
SECURITY_ENTITY *security_entity;
POST_DICTIONARY *post_dictionary;
DICTIONARY_SEPARATE_POST_PROMPT_LOOKUP *
    dictionary_separate;
QUERY_CHART *query_chart;
int row_list_length;
int query_row_list_days_range;
CHART_AGGREGATE_LIST *chart_aggregate_list;
```



CHART_AGGREGATE_LIST

```
/* Usage */
CHART_AGGREGATE_LIST *chart_aggregate_list_new(
    char *date_attribute_name,
    char *time_attribute_name,
    LIST *query_select_name_list,
    LIST *query_fetch_row_list,
    int query_row_list_days_range() );

/* Process */
CHART_AGGREGATE_LIST *chart_aggregate_list_calloc(
    void );

char *chart_aggregate_list_end_date(
    LIST *query_fetch_row_list );

list_rewind( query_select_name_list );

/* Skip date */
list_next( query_select_name_list );

/* Skip time */
if ( time_attribute_name )
{
    list_next( query_select_name_list );
}

LIST *chart_aggregate_attribute_list = list_new();

do {
    char *number_attribute_name =
        list_get(
            query_select_name_list );

    CHART_AGGREGATE_ATTRIBUTE *
        chart_aggregate_attribute_new(
            date_attribute_name,
            number_attribute_name,
            query_fetch_row_list,
            query_row_list_days_range,
            chart_aggregate_list_end_date() );

    list_set(
        chart_aggregate_attribute_list,
        chart_aggregate_attribute_new() );

} while ( list_next( query_select_name_list ) );

/* Attributes */
char *end_date;
LIST *chart_aggregate_attribute_list;
```

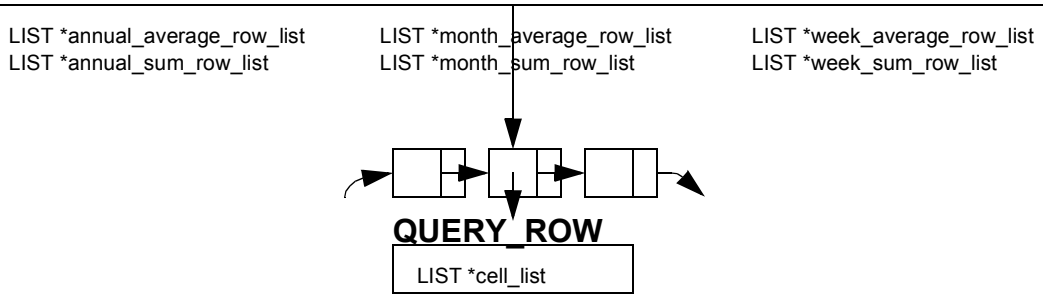


CHART_AGGREGATE_ATTRIBUTE

```
char *attribute_name
LIST *annual_average_row_list
LIST *annual_sum_row_list
LIST *month_average_row_list
LIST *month_sum_row_list
LIST *week_average_row_list
LIST *week_sum_row_list
```

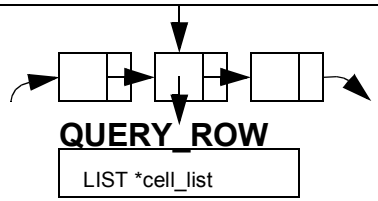
CHART_AGGREGATE_ATTRIBUTE

```
/* Usage */
CHART_AGGREGATE_ATTRIBUTE *
   chart_aggregate_attribute_new(
      char *date_attribute_name,
      char *number_attribute_name,
      LIST *query_fetch_row_list,
      int query_row_list_days_range(),
      char *chart_aggregate_list_end_date() );

/* Process */
CHART_AGGREGATE_ATTRIBUTE *
   chart_aggregate_attribute_calloc(
      void );

if ( query_row_list_days_range > 1825 )
{
   CHART_AGGREGATE *chart_aggregate_new(
      date_attribute_name,
      number_attribute_name,
      query_fetch_row_list,
      chart_aggregate_end_list_date,
      average /* enum aggregate_statistic */,
      annually /* enum aggregate_level */ );

   LIST *annual_average_row_list =
      chart_aggregate_new()->query_row_list;

   LIST *annual_sum_row_list =
      CHART_AGGREGATE *chart_aggregate_new(
         date_attribute_name,
         number_attribute_name,
         query_fetch_row_list,
         chart_aggregate_list_end_date,
         sum /* enum aggregate_statistic */,
         annually /* enum aggregate_level */ );

   LIST *annual_sum_row_list =
      chart_aggregate_new()->query_row_list;
}

if ( query_row_list_days_range > 730 )
{
   CHART_AGGREGATE *chart_aggregate_new(
      date_attribute_name,
      number_attribute_name,
      query_fetch_row_list,
      chart_aggregate_list_end_date,
      average /* enum aggregate_statistic */,
      monthly /* enum aggregate_level */ );

   LIST *month_average_row_list =
      chart_aggregate_new()->query_row_list;

   CHART_AGGREGATE *chart_aggregate_new(
      date_attribute_name,
      number_attribute_name,
      query_fetch_row_list,
      chart_aggregate_list_end_date,
      sum /* enum aggregate_statistic */,
      monthly /* enum aggregate_level */ );

   LIST *month_sum_row_list =
      chart_aggregate_new()->query_row_list;
}

CHART_AGGREGATE *chart_aggregate_new(
   date_attribute_name,
   number_attribute_name,
   query_fetch_row_list,
   chart_aggregate_list_end_date,
   average /* enum aggregate_statistic */,
   weekly /* enum aggregate_level */ );

LIST *week_average_row_list =
   chart_aggregate_new()->query_row_list;

CHART_AGGREGATE *chart_aggregate_new(
   date_attribute_name,
   number_attribute_name,
   query_fetch_row_list,
   chart_aggregate_list_end_date,
   sum /* enum aggregate_statistic */,
   weekly /* enum aggregate_level */ );

LIST *week_sum_row_list =
   chart_aggregate_new()->query_row_list;

/* Attributes */
char *number_attribute_name;
LIST *annual_average_row_list;
LIST *annual_sum_row_list;
LIST *month_average_row_list;
LIST *month_sum_row_list;
LIST *week_average_row_list;
LIST *week_sum_row_list;
```



CHART_AGGREGATE

```
/* Usage */
CHART_AGGREGATE *chart_aggregate_new(
    char *date_attribute_name,
    char *number_attribute_name,
    LIST *query_fetch_row_list,
    char *chart_aggregate_list_end_date(),
    enum aggregate_statistic aggregate_statistic,
    enum aggregate_level aggregate_level );

/* Process */
CHART_AGGREGATE *chart_aggregate_calloc(
    void );

LIST *chart_aggregate_fetch_list(
    query_fetch_row_list,
    number_attribute_name,
    chart_aggregate_list_end_date,
    aggregate_statistic,
    aggregate_level );

LIST *query_row_list =
    LIST *chart_aggregate_query_row_list(
        date_attribute_name,
        number_attribute_name,
        chart_aggregate_fetch_list() );

/* Usage */
LIST *chart_aggregate_query_row_list(
    char *date_attribute_name,
    char *numer_attribute_name,
    LIST *chart_aggregate_fetch_list );

/* Process */
LIST *query_row_list = list_new();

for CHART_AGGREGATE_FETCH *
    chart_aggregate_fetch in
        chart_aggregate_fetch_list
{
    LIST *chart_aggregate_query_cell_list(
        date_attribute_name,
        number_attribute_name,
        chart_aggregate_fetch->date_string,
        char_aggregate_fetch->value_string );

    QUERY_ROW *query_row_new(
        0 /* not input_save_boolean */,
        (char *)0 /* input */,
        chart_aggregate_query_cell_list() );

    list_set(
        query_row_list,
        query_row_new() );
}

/* Usage */
LIST *chart_aggregate_query_cell_list(
    char *date_attribute_name,
    char *number_attribute_name,
    char *date_string,
    char *value_string );

/* Process */
LIST *query_cell_list = list_new();

list_set(
    query_cell_list,
    QUERY_CELL *query_cell_new(
        date_attribute_name,
        date_string /* select_datum */,
        0 /* primary_key_index,
        1 /* attribute_is_date */,
        0 /* not attribute_is_date_time */,
        0 /* not attribute_is_current_date_time */,
        0 /* not attribute_is_number */,
        (char *)0 /* display_datum */ ) );

list_set(
    query_cell_list,
    QUERY_CELL *query_cell_new(
        number_attribute_name,
        value_string /* select_datum */,
        0 /* primary_key_index */,
        0 /* not attribute_is_date */,
        0 /* not attribute_is_date_time */,
        0 /* not attribute_is_current_date_time */,
        1 /* attribute_is_number */,
        (char *)0 /* display_datum */ ) );

/* Attribute */
LIST *query_row_list;
```

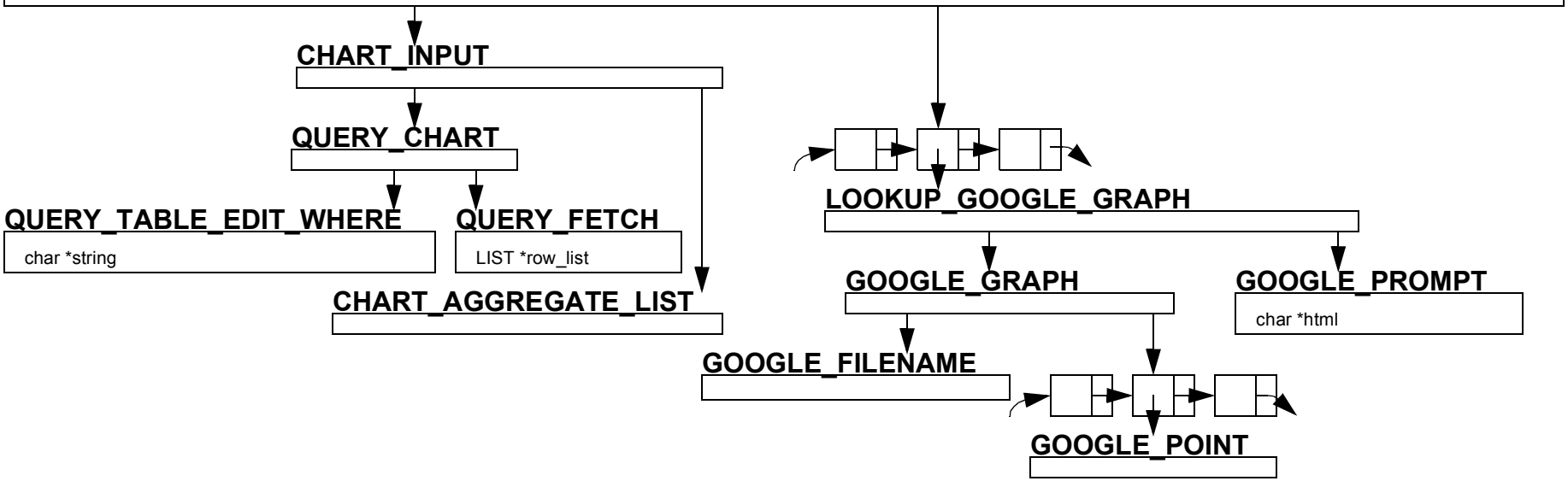


CHART_AGGREGATE_FETCH

```
/* Usage */
LIST *chart_aggregate_fetch_list(
    LIST *query_fetch_row_list,
    char *attribute_name,
    char *chart_aggregate_list_end_date(),
    enum aggregate_statistic aggregate_statistic,
    enum aggregate_level aggregate_level );

/* Process */
char *chart_aggregate_fetch_system_string(
    chart_aggregate_list_end_date,
    aggregate_statistic,
    aggregate_level );

SPOOL *spool_new(
    chart_aggregate_fetch_system_string(),
    0 /* not capture_stderr_boolean */ );

for QUERY_ROW *query_row in query_fetch_row_list
{
    char *chart_aggregate_fetch_output_string(
        attribute_name,
        query_row->cell_list );

    fprintf(
        spool_new()->output_pipe,
        "%s\n",
        chart_aggregate_fetch_output_string() );
}

pclose( spool_new()->output_pipe );

LIST *spool_list(
    spool_new()->output_filespecification );

LIST *fetch_list = list_new();

for char *input_string in spool_list()
{
    CHART_AGGREGATE_FETCH *
        chart_aggregate_fetch =
            chart_aggregate_fetch_parse(
                input_string );

    list_set(
        fetch_list,
        chart_aggregate_fetch );
}

/* Usage */
CHART_AGGREGATE_FETCH *
    chart_aggregate_fetch_parse(
        char *input_string );

/* Process */
piece( date_string[], input_string, 0 );
piece( value_string[], input_string, 1 );

CHART_AGGREGATE_FETCH *
    chart_aggregate_fetch_new(
        strdup( date_string ),
        strdup( value_string ) );

/* Usage */
CHART_AGGREGATE_FETCH *
    chart_aggregate_fetch_new(
        char *date_string,
        char *value_string );

/* Process */
CHART_AGGREGATE_FETCH *
    chart_aggregate_fetch_calloc(
        void );

/* Usage */
char *chart_aggregate_fetch_system_string(
    char *chart_aggregate_list_end_date(),
    enum aggregate_statistic aggregate_statistic,
    enum aggregate_level aggregate_level );

/* Process */
sprintf(
system_string[],
"real_time2aggregate_value.e %s 0 -1 1 '%c' %s '' %s",
    char *aggregate_statistic_string(
        aggregate_statistic ),
    SQL_DELIMITER,
    char *aggregate_level_string(
        aggregate_level ),
    chart_aggregate_list_end_date );

/* Usage */
char *chart_aggregate_fetch_output_string(
    char *attribute_name,
    LIST *cell_list );

/* Process */
QUERY_CELL *first_query_cell =
    list_first( cell_list );

char *date_string = first_query_cell->select_datum;

QUERY_CELL *query_cell_seek(
    attribute_name,
    cell_list );

char *value_string = query_cell_seek()->select_datum;

sprintf(
    static output_string[],
    "%s%c%s",
    date_string,
    SQL_DELIMITER,
    value_string );

/* Attributes */
char *date_string;
char *value_string;
```

LOOKUP_GOOGLE (1)

```
/* Usage */
LOOKUP_GOOGLE *lookup_google_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *dictionary_string,
    char *data_directory );

/* Process */
LOOKUP_GOOGLE *lookup_google_calloc( void );

CHART_INPUT *chart_input =
    chart_input_new(
        application_name,
        login_name,
        role_name,
        folder_name,
        dictionary_string );

if ( !chart_input->row_list_length ) return this;

LIST *lookup_google_graph_list = list_new();

int chart_number = 0;

for char *number_attribute_name
    chart_input->
        query_chart->
        number_attribute_name_list )
{
    char *chart_title(
        folder_name,
        (char *)0 /* number_attribute_name */,
        aggregate_statistic_none,
        aggregate_level_none,
        chart_input->
            query_chart->
            query_table_edit_where->
            string );

    char *chart_filename_key(
        GOOGLE_FILENAME_STEM,
        number_attribute_name,
        aggregate_statistic_none,
        aggregate_level_none );

LOOKUP_GOOGLE_GRAPH *lookup_google_graph =
    lookup_google_graph_new(
        application_name,
        session_key,
        data_directory,
        chart_input->
            query_chart->
            date_attribute_name,
        chart_input->
            query_chart->
            time_attribute_name,
        chart_input->
            query_chart->
            query_fetch->
            row_list,
        number_attribute_name,
        chart_title(),
        chart_filename_key(),
        ++chart_number );

    list_set(
        lookup_google_graph_list,
        lookup_google_graph );
}
```

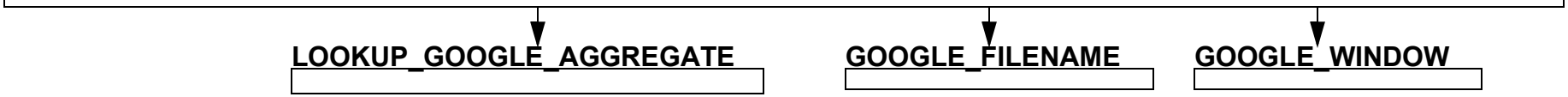


LOOKUP_GOOGLE (2)

```
/* Process (continued) */
if ( chart_input->chart_aggregate_list )
{
    LOOKUP_GOOGLE_AGGREGATE *
        lookup_google_aggregate =
            lookup_google_aggregate_new(
                application_name,
                session_key,
                folder_name,
                data_directory,
                chart_input->query_chart,
                chart_input->
                    chart_aggregate_list->
                    chart_aggregate_attribute_list,
                chart_number );

    list_set_list(
        lookup_google_graph_list,
        lookup_google_aggregate->
            google_graph_list );
}

GOOGLE_FILENAME *lookup_google_filename(
    lookup_google_graph_list );

GOOGLE_WINDOW *google_window =
    google_window_new(
        lookup_google_filename(),
        chart_input>
            query_chart->
            query_table_edit_where->
            string /* sub_title */,
        getpid() /* process_id */ );
```

```
/* Usage */
GOOGLE_FILENAME *lookup_google_filename(
    LIST *lookup_google_graph_list );

/* Process */
if ( list_length( lookup_google_graph_list ) == 1 )
{
    LOOKUP_GOOGLE_GRAPH *lookup_google_graph =
        list_first(
            lookup_google_graph_list );

    return
    lookup_google_graph->
        google_graph->
        google_filename;
}
else
{
    return NULL;
}

/* Driver */
void document_head_output_html_stream( stdout );

printf(
    "%s\n",
    lookup_google->
        google_window->
        html );

printf( "<table border=1>\n" );
```

```
printf( "<tr><th>Chart\n" );

for LOOKUP_GOOGLE_GRAPH *lookup_google_graph
    in lookup_google->lookup_google_graph_list
{
    FILE *appaserver_output_file(
        char *lookup_google_graph->
            google_graph->
            google_filename->
            http_output_filename );

    google_graph_output(
        lookup_google_graph->google_graph,
        appaserver_output_file() );

    fclose( appaserver_output_file() );

    printf(
        "%s\n",
        lookup_google_graph->
            google_prompt->
            html );
}

printf( "</table>\n" );
document_close();

/* Attributes */
CHART_INPUT *chart_input;
LIST *lookup_google_graph_list;
LOOKUP_GOOGLE_AGGREGATE *
    lookup_google_aggregate;
GOOGLE_FILENAME *google_filename;
GOOGLE_WINDOW *google_window;
```



LOOKUP_GOOGLE_AGGREGATE (1)

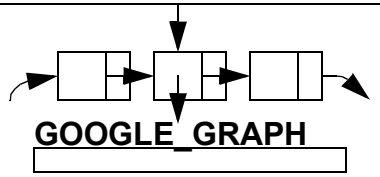
```
/* Usage */
LOOKUP_GOOGLE_AGGREGATE *
    lookup_google_aggregate_new(
        char *application_name,
        char *session_key,
        char *folder_name,
        char *data_directory,
        QUERY_CHART *query_chart,
        LIST *chart_aggregate_attribute_list,
        int chart_number );

/* Process */
LOOKUP_GOOGLE_AGGREGATE *
    lookup_google_aggregate_calloc(
        void );

LIST *google_graph_list = list_new();

for CHART_AGGREGATE_ATTRIBUTE *
    chart_aggregate_attribute in
        chart_aggregate_attribute_list
{
    if ( list_length(
            chart_aggregate_attribute->
                week_average_row_list ) )
    {
        char *chart_title(
            folder_name,
            (char *)0 /* number_attribute_name */,
            average /* enum aggregate_statistic */,
            weekly /* enum aggregate_level */,
            query_chart->
                query_table_edit_where->
                    string );

        char *chart_filename_key(
            GOOGLE_FILENAME_STEM,
            chart_aggregate_attribute->
                number_attribute_name,
            average /* enum aggregate_statistic */,
            weekly /* enum aggregate_level */ );

        LOOKUP_GOOGLE_GRAPH *lookup_google_graph =
            lookup_google_graph_new(
                application_name,
                session_key,
                data_directory,
                query_chart->
                    date_attribute_name,
                (char *)0 /* time_attribute_name */,
                chart_aggregate_attribute->
                    week_average_row_list,
                chart_aggregate_attribute->
                    number_attribute_name,
                chart_title(),
                chart_filename_key(),
                ++chart_number );

        list_set(
            lookup_google_graph_list,
            lookup_google_graph );

        char *chart_title(
            folder_name,
            (char *)0 /* number_attribute_name */,
            sum /* enum aggregate_statistic */,
            weekly /* enum aggregate_level */,
            query_chart->
                query_table_edit_where->
                    string );

        char *chart_filename_key(
            GOOGLE_FILENAME_STEM,
            chart_aggregate_attribute->
                number_attribute_name,
            sum /* enum aggregate_statistic */,
            weekly /* enum aggregate_level */ );

        LOOKUP_GOOGLE_GRAPH *lookup_google_graph =
            lookup_google_graph_new(
                application_name,
                session_key,
                data_directory,
                query_chart->
                    date_attribute_name,
                (char *)0 /* time_attribute_name */,
                chart_aggregate_attribute->
                    week_sum_row_list,
                chart_aggregate_attribute->
                    number_attribute_name,
                chart_title(),
                chart_filename_key(),
                ++chart_number );

        list_set(
            lookup_google_graph_list,
            lookup_google_graph );
    } /* if week_average_row_list */
```

LOOKUP_GOOGLE_AGGREGATE (2)

```
/* Process (continued) */
    if ( list_length(
            chart_aggregate_attribute->
               month_average_row_list ) )
    {
        char *chart_title(
            folder_name,
            (char *)0 /* number_attribute_name */,
            average /* enum aggregate_statistic */,
            monthly /* enum aggregate_level */,
            query_chart->
               query_table_edit_where->
               string );

        char *chart_filename_key(
            GOOGLE_FILENAME_STEM,
            chart_aggregate_attribute->
               number_attribute_name,
            average /* enum aggregate_statistic */,
            monthly /* enum aggregate_level */ );

        LOOKUP_GOOGLE_GRAPH *lookup_google_graph =
            lookup_google_graph_new(
               application_name,
               session_key,
               data_directory,
               query_chart->
                  date_attribute_name,
               (char *)0 /* time_attribute_name */,
               chart_aggregate_attribute->
                  month_average_row_list,
               chart_aggregate_attribute->
                  number_attribute_name,
               chart_title(),
               chart_filename_key(),
               ++chart_number );

        list_set(
            lookup_google_graph_list,
            lookup_google_graph );

        char *chart_title(
            folder_name,
            (char *)0 /* number_attribute_name */,
            sum /* enum aggregate_statistic */,
            monthly /* enum aggregate_level */,
            query_chart->
               query_table_edit_where->
               string );

        char *chart_filename_key(
            GOOGLE_FILENAME_STEM,
               chart_aggregate_attribute->
                  number_attribute_name,
               sum /* enum aggregate_statistic */,
               monthly /* enum aggregate_level */ );

            LOOKUP_GOOGLE_GRAPH *lookup_google_graph =
               lookup_google_graph_new(
                  application_name,
                  session_key,
                  data_directory,
                  query_chart->
                     date_attribute_name,
                  (char *)0 /* time_attribute_name */,
                  chart_aggregate_attribute->
                     month_sum_row_list,
                  chart_aggregate_attribute->
                     number_attribute_name,
                  chart_title(),
                  chart_filename_key(),
                  ++chart_number );

            list_set(
               lookup_google_graph_list,
               lookup_google_graph );
        } /* if month_average_row_list */
```

LOOKUP_GOOGLE_AGGREGATE (3)

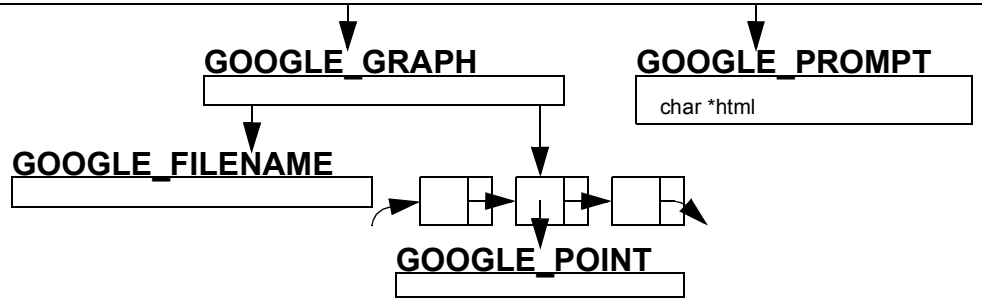
```
/* Process (continued) */
    if ( list_length(
            chart_aggregate_attribute->
               annual_average_row_list ) )
    {
        char *chart_title(
            folder_name,
            (char *)0 /* number_attribute_name */,
            average /* enum aggregate_statistic */,
            annually /* enum aggregate_level */,
            query_chart->
               query_table_edit_where->
               string );

        char *chart_filename_key(
            GOOGLE_FILENAME_STEM,
            chart_aggregate_attribute->
               number_attribute_name,
            average /* enum aggregate_statistic */,
            annually /* enum aggregate_level */ );

        LOOKUP_GOOGLE_GRAPH *lookup_google_graph =
            lookup_google_graph_new(
               application_name,
               session_key,
               data_directory,
               query_chart->
                  date_attribute_name,
               (char *)0 /* time_attribute_name */,
               chart_aggregate_attribute->
                  annual_average_row_list,
               chart_aggregate_attribute->
                  number_attribute_name,
               chart_title(),
               chart_filename_key(),
               ++chart_number );

        list_set(
            lookup_google_graph_list,
            lookup_google_graph );

        char *chart_title(
            folder_name,
            (char *)0 /* number_attribute_name */,
            sum /* enum aggregate_statistic */,
            annually /* enum aggregate_level */,
            query_chart->
               query_table_edit_where->
               string );

        char *chart_filename_key(
            GOOGLE_FILENAME_STEM,
            chart_aggregate_attribute->
               number_attribute_name,
               sum /* enum aggregate_statistic */,
               annually /* enum aggregate_level */ );

        LOOKUP_GOOGLE_GRAPH *lookup_google_graph =
            lookup_google_graph_new(
               application_name,
               session_key,
               data_directory,
               query_chart->
                  date_attribute_name,
               (char *)0 /* time_attribute_name */,
               chart_aggregate_attribute->
                  annual_sum_row_list,
               chart_aggregate_attribute->
                  number_attribute_name,
               chart_title(),
               chart_filename_stem(),
               ++chart_number );

            list_set(
               lookup_google_graph_list,
               lookup_google_graph );
        } /* if annual_average_row_list */
    } /* for each chart_aggregate_attribute_list */

    /* Attribute */
    LIST *google_graph_list;
```



LOOKUP_GOOGLE_GRAPH

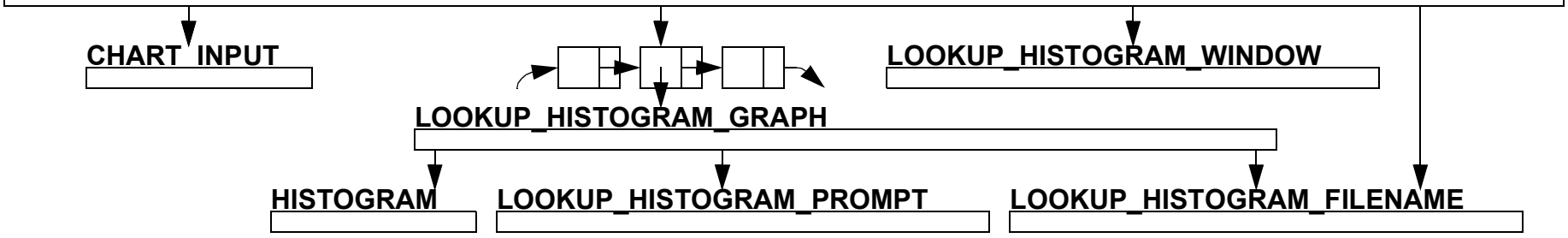
```
/* Usage */
LOOKUP_GOOGLE_GRAPH *
    lookup_google_graph_new(
        char *application_name,
        char *session_key,
        char *data_directory,
        char *date_attribute_name,
        char *time_attribute_name,
        LIST *query_fetch_row_list,
        char *number_attribute_name,
        char *chart_title,
        char *chart_filename_key,
        int chart_number );
```

```
/* Process */
LOOKUP_GOOGLE_GRAPH *
    lookup_google_graph_calloc(
        void );

GOOGLE_GRAPH *google_graph =
    google_graph_new(
        application_name,
        session_key,
        data_directory,
        date_attribute_name,
        time_attribute_name,
        number_attribute_name,
        query_fetch_row_list,
        chart_title,
        chart_filename_key,
        chart_number,
        "ColumnChart" /* google_graph_function */ );

GOOGLE_PROMPT *google_prompt =
    google_prompt_new(
        google_graph->google_filename,
        chart_filename_key,
        getpid() /* process_id */ );

/* Attributes */
GOOGLE_GRAPH *google_graph;
GOOGLE_PROMPT *google_prompt;
```



LOOKUP_HISTOGRAM

```
/* Usage */
LOOKUP_HISTOGRAM *lookup_histogram_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *dictionary_string,
    char *data_directory );

/* Process */
LOOKUP_HISTOGRAM *lookup_histogram_calloc(
    void );

CHART_INPUT *chart_input =
    chart_input_new(
        application_name,
        login_name,
        role_name,
        folder_name,
        dictionary_string );

if ( !chart_input->row_list_length ) return this;

if ( !list_rewind(
        chart_input->
            query_chart->
            number_attribute_name_list ) )
{
    exit( 1 );
}

LIST *lookup_histogram_graph_list = list_new();

for char *number_attribute_name
    in chart_input->
            query_chart->
            number_attribute_name_list
{
    LOOKUP_HISTOGRAM_GRAPH *
        lookup_histogram_graph =
            lookup_histogram_graph_new(
                application_name,
                session_key,
                folder_name,
                data_directory,
                chart_input->
                    query_chart->
                    query_edit_table_where->
                    string /* where_string */,
                chart_input->
                    query_chart->
                    query_fetch->
                    row_list,
                number_attribute_name );

    list_set(
        lookup_histogram_graph_list,
        lookup_histogram_graph );
}

LOOKUP_HISTOGRAM_FILENAME *
    lookup_histogram_filename(
        lookup_histogram_graph_list );

LOOKUP_HISTOGRAM_WINDOW *
    lookup_histogram_window =
        lookup_histogram_window_new(
            folder_name,
            lookup_histogram_filename(),
            chart_input->
                query_chart->
                query_edit_table_where->
                string /* where_string */,
            getpid() /* process_id */ );

/* Driver */
void document_head_output_html_stream( stdout );

printf(
    "%s\n",
    lookup_histogram->
        lookup_histogram_window->
            html );

for LOOKUP_HISTOGRAM_GRAPH *
    lookup_histogram_graph
        in lookup_histogram->
        lookup_histogram_graph_list
{
    void histogram_pdf_create(
        lookup_histogram_graph->
            number_list,
        lookup_histogram_graph->
            histogram->
            system_string );

    printf(
        "%s\n",
        lookup_histogram_graph->
            lookup_histogram_prompt->
            html );
}

document_close();

/* Attributes */
CHART_INPUT *chart_input;
LIST *lookup_histogram_graph_list;
LOOKUP_HISTOGRAM_FILENAME *
    lookup_histogram_filename;
LOOKUP_HISTOGRAM_WINDOW *
    lookup_histogram_window;
```



LOOKUP_HISTOGRAM_WINDOW

```
/* Usage */
LOOKUP_HISTOGRAM_WINDOW *
    lookup_histogram_window_new(
        char *folder_name,
        LOOKUP_HISTOGRAM_FILENAME *
            lookup_histogram_filename(),
        char *where_string,
        pid_t process_id );

/* Process */
LOOKUP_HISTOGRAM_WINDOW *
    lookup_histogram_window_calloc(
        void );

if ( lookup_histogram_filename )
{
    char *query_prompt_target_frame(
        (char *)0 /* datatype_name */,
        process_id );

    char *lookup_histogram_window_onload_string(
        char *lookup_histogram_filename->
            appaserver_link->
            appaserver_link_prompt->
            filename
                /* http_prompt_filename */,
        char *query_prompt_target_frame() );
}

char *document_body_onload_open_tag(
    JAVASCRIPT_WAIT_OVER,
    lookup_histogram_window_onload_string()
        /* javascript_replace */,
    (char *)0 /* menu_onload_string */ );

char *date_time_now( date_utc_offset() );

char *lookup_histogram_window_title_html(
        char *folder_name,
        char *date_time_now() );

char *query_where_display(
    where_string,
    0 /* max_length */ );

char *lookup_histogram_window_html(
    char *document_body_onload_open_tag(),
    char *lookup_histogram_window_title_html(),
    char *query_where_display() );

free( document_onload_body_open_tag() );
free( date_time_now() );
free( query_where_display() );

/* Attribute */
char *html;
```

LOOKUP_HISTOGRAM_GRAPH

```
/* Usage */
LOOKUP_HISTOGRAM_GRAPH *
   lookup_histogram_graph_new(
      char *application_name,
      char *session_key,
      char *folder_name,
      char *data_directory,
      char *where_string,
      LIST *query_fetch_row_list,
      char *number_attribute_name );

/* Process */
LOOKUP_HISTOGRAM_GRAPH *
   lookup_histogram_graph_calloc(
      void );

char *lookup_histogram_graph_title(
   char *folder_name,
   char *number_attribute_name );
```

```
LOOKUP_HISTOGRAM_FILENAME *
   lookup_histogram_filename =
      lookup_histogram_filename_new(
         application_name,
         session_key,
         data_directory,
         number_attribute_name );

HISTOGRAM *histogram =
   histogram_new(
      query_fetch_row_list,
      number_attribute_name,
      lookup_histogram_graph_title(),
      where_string,
      lookup_histogram_filename->
         appaserver_link->
         appaserver_link_output->
         filename
            /* histogram_output_pdf */ );
```

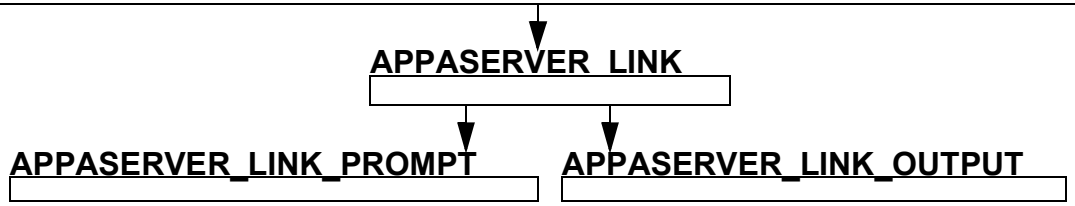
```
LOOKUP_HISTOGRAM_PROMPT *
   lookup_histogram_prompt =
      lookup_histogram_prompt_new(
         lookup_histogram_filename->
            appaserver_link->
            appaserver_link_prompt->
            filename
               /* http_prompt_filename */,
         number_attribute_name,
         getpid() /* process_id */ );

/* Attributes */
char *title;
LOOKUP_HISTOGRAM_FILENAME *
   lookup_histogram_filename;
HISTOGRAM *histogram;
LOOKUP_HISTOGRAM_PROMPT *
   lookup_histogram_prompt;
```

LOOKUP_HISTOGRAM_PROMPT

```
/* Usage */
LOOKUP_HISTOGRAM_PROMPT *
    lookup_histogram_prompt_new(
        char *http_prompt_filename,
        char *number_attribute_name,
        pid_t process_id );

/* Process */
LOOKUP_HISTOGRAM_PROMPT *
    lookup_histogram_prompt_calloc(
        void );

char *query_prompt_target_frame(
    number_attribute_name
        /* datatype_name */,
    process_id );

char *lookup_histogram_prompt_html(
    http_prompt_filename,
    number_attribute_name,
    query_prompt_target_frame() );

/* Usage */
char *lookup_histogram_prompt_html(
    char *http_prompt_filename,
    char *number_attribute_name,
    char *query_prompt_target_frame() );

/* Process */
char html[];
char *ptr = html;

ptr += sprintf( ptr,
    "<br><br>" );

ptr += sprintf( ptr,
    "<a href=\"%s\" target=%s>Press to view ",
    http_prompt_filename,
    query_prompt_target_frame );

if ( number_attribute_name )
{
    ptr += sprintf( ptr,
        "%s ",
        string_initial_capital(
            buffer[],
            number_attribute_name ) );
}

sprintf( ptr, "histogram.</a>" );

/* Attributes */
char *html;
```

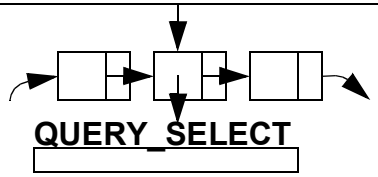
LOOKUP_HISTOGRAM_FILENAME

```
/* Usage */
LOOKUP_HISTOGRAM_FILENAME *
   lookup_histogram_filename_new(
      char *application_name,
      char *session_key,
      char *data_directory,
      char *number_attribute_name );

/* Process */
LOOKUP_HISTOGRAM_FILENAME *
   lookup_histogram_filename_calloc(
      void );

APPASERVER_LINK *appaserver_link =
   appaserver_link_new(
      application_http_prefix(
         application_ssl_support_boolean(
            application_name ) ),
      application_server_address(),
      data_directory,
      LOOKUP_HISTOGRAM_FILENAME_STEM,
      application_name,
      0 /* process_id */,
      session_key,
      number_attribute_name
         /* begin_date_string */,
      (char *)0 /* end_date_string */,
      "pdf" /* extension */ );

/* Usage */
LOOKUP_HISTOGRAM_FILENAME *
   lookup_histogram_filename(
      LIST *lookup_histogram_graph_list );

/* Process */
if ( list_length( lookup_histogram_graph_list ) == 1 )
{
   LOOKUP_HISTOGRAM_GRAPH *
      lookup_histogram_graph =
         list_first(
            lookup_histogram_graph_list );

   return
   lookup_histogram_graph->
      lookup_histogram_filename;
}
else
{
   return NULL;
}

/* Attributes */
APPASERVER_LINK *appaserver_link;
```



QUERY_GROUP (1)

```
/* Usage */
QUERY_GROUP *query_group_new(
    char *application_name,
    char *folder_name,
    char *query_table_edit_where_string,
    LIST *relation_mto1_isa_list,
    LIST *folder_attribute_float_name_list(),
    unsigned long group_count_row_count(),
    LIST *relation_mto1_non_isa->foreign_key_list );

/* Process */
QUERY_GROUP *query_group_calloc( void );

LIST *query_group_select_list(
    application_name,
    folder_name,
    folder_attribute_float_name_list,
    group_count_row_count,
    foreign_key_list,
    list_length( relation_mto1_isa_list )
        /* relation_mto1_isa_list_length */ );

char *query_select_list_string(
    query_group_select_list(),
    (LIST *)0 /* common_name_list */ );

char *query_from_string(
    folder_name,
    relation_mto1_isa_list,
    (char *)0 /* row_security_role_update_list_from */ );

char *query_group_by_string(
    LIST *foreign_key_list );

char *query_group_order_string();

char *query_system_string(
    application_name,
    query_select_list_string(),
    query_from_string(),
    query_table_edit_where_string,
    query_group_by_string(),
    query_group_order_string(),
    0 /* max_rows */ );

/* Attributes */
LIST *select_list;
char *query_select_list_string;
char *query_from_string;
char *query_group_by_string;
char *order_string;
char *query_system_string;
```



QUERY_GROUP (2)

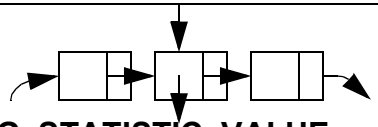
```
/* Usage */
LIST *query_group_select_list(
    char *application_name,
    char *folder_name,
    LIST *folder_attribute_float_name_list,
    unsigned long group_count_row_count,
    LIST *foreign_key_list,
    int relation_mto1_isa_list_length );

/* Process */
LIST *select_list = list_new();

for char *foreign_key in foreign_key_list
{
    list_set(
        select_list,
        QUERY_SELECT *query_select_new(
            relation_mto1_isa_list_length,
            (char *)0 /* attribute_not_null */,
            folder_name,
            foreign_key /* attribute_name */ ) );
}

list_set(
    select_list,
    QUERY_SELECT *query_select_new(
        0 /* relation_mto1_isa_list_length */,
        (char *)0 /* attribute_not_null */,
        (char *)0 /* folder_name */,
        "count(1)" /* attribute_name */ ) );

char *query_group_percent_expression(
    group_count_row_count );

list_set(
    select_list,
    QUERY_SELECT *query_select_new(
        0 /* relation_mto1_isa_list_length */,
        (char *)0 /* attribute_not_null */,
        (char *)0 /* folder_name */,
        query_group_percent_expresssion() ) );

for char *float_attribute_name in
    folder_attribute_float_name_list
{
    char *query_group_sum_expression(
        float_attribute_name );

    list_set(
        select_list,
        QUERY_SELECT *query_select_new(
            0 /* relation_mto1_isa_list_length */,
            (char *)0 /* attribute_not_null */,
            (char *)0 /* folder_name */,
            query_group_sum_expresssion() ) );

    char *query_group_average_expression(
        float_attribute_name );

    list_set(
        select_list,
        QUERY_SELECT *query_select_new(
            0 /* relation_mto1_isa_list_length */,
            (char *)0 /* attribute_not_null */,
            (char *)0 /* folder_name */,
            query_group_average_expresssion() ) );
}

/* Usage */
char *query_group_order_string( void );

/* Process */
return "count(1) desc"

/* Usage */
char *query_group_percent_expression(
    unsigned long group_count_row_count );

/* Process */
snprintf(   char expression[],
            sizeof ( expression ),
            "(count(1) / %ld) * 100.0",
            group_count_row_count );

/* Usage */
char *query_group_sum_expression(
    char *float_attribute_name );

/* Process */
snprintf(   char expression[],
            sizeof ( expression ),
            "sum(%s)",
            float_attribute_name );

/* Usage */
char *query_group_average_expression(
    char *float_attribute_name );

/* Process */
snprintf(   char expression[],
            sizeof ( expression ),
            "avg(%s)",
            float_attribute_name );
```

MOVING_STATISTIC (1)

```
/* Usage */
MOVING_STATISTIC *moving_statistic_new(
    int days_count,
    int value_piece,
    char delimiter,
    enum aggregate_statistic aggregate_statistic,
    char *begin_date_string );

/* Process */
MOVING_STATISTIC *moving_statistic_calloc( void );

moving_statistic_value_list = list_new();

/* Usage */
void moving_statistic_input(
    LIST *moving_statistic_value_list,
    char delimiter,
    char *strdup( input[] ) );

/* Process */
piece( date_string[], delimiter, input, 0 );

list_set(
    moving_statistic_value_list,
    moving_statistic_value_new(
        input,
        strdup( date_string ) ) );

/* Usage */
void moving_statistic_display(
    int days_count,
    int value_piece,
    char delimiter,
    enum aggregate_statistic aggregate_statistic,
    char *begin_date_string,
    LIST *moving_statistic_value_list );

/* Process */
FILE *sort_pipe = popen( "sort", "w" );

list_tail( moving_statistic_value_list );

do {
    MOVING_STATISTIC_VALUE *
        moving_statistic_value =
            list_get(
                moving_statistic_value_list );

    STATISTIC *statistic =
        statistic_new(
            value_piece,
            -1 /* weight_piece */,
            delimiter );

    list_push( moving_statistic_value_list );

    moving_statistic_accumulate(
        statistic,
        days_count,
        moving_statistic_value_list );

    list_pop( moving_statistic_value_list );

    statistic->statistic_calculate =
        statistic_calculate_new(
            statistic->statistic_number_array,
            statistic->weight_count,
            statistic->sum,
            statistic->row_count,
            statistic->value_count,
            statistic->non_zero_count );

    statistic_number_array_sort(
        statistic->statistic_number_array /* in/out */ );

    statistic->statistic_quartile =
        statistic_quartile_new(
            statistic->statistic_number_array );

    moving_statistic_output(
        value_piece,
        delimiter,
        aggregate_statistic,
        sort_pipe,
        moving_statistic_value->input,
        statistic->sum,
        statistic->statistic_calculate,
        statistic->statistic_quartile );

    statistic_free( statistic );

    if ( strcmp(
            moving_statistic_value->date_string,
            begin_date_string ) <= 0 )
    {
        break;
    }
} while ( list_previous( moving_statistic_value_list ) );

pclose( sort_pipe );
```

MOVING_STATISTIC (2)

```
/* Usage */
void moving_statistic_accumulate(
    STATISTIC *statistic /* in/out */,
    int days_count,
    LIST *moving_statistic_value_list );

/* Process */
while ( 1 )
{
    MOVING_STATISTIC_VALUE *
        moving_statistic_value =
            list_get(
                moving_statistic_value_list );

    statistic_accumulate(
        statistic /* in/out */,
        statistic->statistic_number_array /* in/out */,
        moving_statistic_value->input );

    list_previous( moving_statistic_value_list );
    if ( --days_count == 0 ) break;
}

/* Usage */
void moving_statistic_output(
    int value_piece,
    char delimiter,
    enum aggregate_statistic aggregate_statistic,
    FILE *sort_pipe,
    char *input,
    double statistic->sum,
    STATISTIC_CALCULATE *statistic_calculate,
    STATISTIC_QUARTILE *statistic_quartile );

/* Process */
char *moving_statistic_value_string(
    enum aggregate_statistic aggregate_statistic,
    double sum,
    STATISTIC_CALCULATE *statistic_calculate,
    STATISTIC_QUARTILE *statistic_quartile );

char *moving_statistic_replace(
    int value_piece,
    char delimiter,
    char *input,
    char *moving_statistic_value_string() );

fprintf(
    sort_pipe,
    "%s\n",
    moving_statistic_replace() );

free( moving_statistic_replace() );

/* Attributes */
int days_count;
int value_piece;
char delimiter;
enum aggregate_statistic aggregate_statistic;
char *begin_date_string;
LIST *moving_statistic_value_list;
```



MOVING_STATISTIC_VALUE

```
/* Usage */
MOVING_STATISTIC_VALUE *
    moving_statistic_value_new(
        char *input,
        char *date_string );

/* Process */
MOVING_STATISTIC_VALUE *
    moving_statistic_value_calloc(
        void );

/* Attributes */
char *input;
char *date_string;
```

STATISTIC (1)

```
/* Usage */
STATISTIC *statistic_new(
    int value_piece,
    int weight_piece,
    char delimiter );

/* Process */
STATISTIC *statistic_calloc( void );

STATISTIC_NUMBER_ARRAY *
    statistic_number_array =
        statistic_number_array_calloc();

/* Usage */
void statistic_accumulate(
    STATISTIC *statistic
        /* in/out */,
    STATISTIC_NUMBER_ARRAY *statistic_number_array
        /* in/out */,
    char *input );

/* Process */
unsigned long row_count++;

boolean statistic_empty_value(
    int statistic->value_piece,
    char statistic->delimiter,
    char *input );

if ( !statistic_empty_value() )
{
    double statistic_value(
        int statistic->value_piece,
        char statistic->delimiter,
        char *input );

unsigned long value_count++;

/* Returns at least one */
int statistic_weight(
    int statistic->weight_piece,
    char statistic->delimiter,
    char *input );

unsigned long weight_count += statistic_weight();

double statistic_weight_value(
    double statistic_value(),
    int statistic_weight() );

double sum += statistic_weight_value();

if (    statistic_number_array
&&  !statistic_number_array->
        max_numbers_exceeded )
{
    void statistic_number_array_insert(
        statistic_number_array /* in/out */,
        STATISTIC_NUMBER_ARRAY_MAX_SIZE,
        statistic_value(),
        statistic_weight() );
}

if ( statistic_weight_value() )
{
    unsigned long non_zero_count++;
}

} /* if !statistic_empty_value() */

/* Usage */
void statistic_free(
    STATISTIC *statistic );

/* Process */
statistic_number_array_free(
    statistic->statistic_number_array );

if ( statistic->statistic_calculate )
{
    free( statistic->statistic_calculate );
}

free( statistic );

/* Attributes */
int value_piece;
int weight_piece;
char delimiter;
unsigned long row_count;
boolean empty_value;
double value;
unsigned long value_count;
int weight;
unsigned long weight_count;
double weight_value;
double sum;
STATISTIC_NUMBER_ARRAY *statistic_number_array;
unsigned long non_zero_count;
STATISTIC_CALCULATE *statistic_calculate;
STATISTIC_QUARTILE *statistic_quartile;
```

STATISTIC (2)

```
/* Usage */
LIST *statistic_table_cell_list(
    STATISTIC *statistic,
    char *label,
    boolean outlier_boolean );

/* Process */
LIST *cell_list = list_new();

list_set(
     cell_list,
     HTML_CELL *html_cell_new(
          label /* data */,
          0 /* not large_boolean */,
          0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    HTML_CELL *html_cell_new(
        strdup(
            string_commas_unsigned_long(
                statistic->value_count ) ),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    HTML_CELL *html_cell_new(
        strdup(
            string_commas_money(
                statistic->statistic_calculate->sum ) ),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    HTML_CELL *html_cell_new(
        string_commas_money(
            statistic->statistic_quartile->minimum ),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    HTML_CELL *html_cell_new(
        string_commas_money(
            statistic->statistic_calculate->first ),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    HTML_CELL *html_cell_new(
        string_commas_money(
            statistic->statistic_calculate->average ),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    HTML_CELL *html_cell_new(
        string_commas_money(
            statistic->statistic_quartile->median ),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    HTML_CELL *html_cell_new(
        string_commas_money(
            statistic->statistic_calculate->third ),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    HTML_CELL *html_cell_new(
        string_commas_money(
            statistic->statistic_quartile->maximum ),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    HTML_CELL *html_cell_new(
        statistic_quartile_range_string(
            statistic->statistic_quartile->range ),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    HTML_CELL *html_cell_new(
        statistic_quartile_range_string(
            statistic->
                statistic_quartile->
                innerquartile_range ),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    HTML_CELL *html_cell_new(
        string_commas_money(
        statistic->
            statistic_quartile->
            coefficient_of_dispersion ),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

if (    outlier_boolean
&&  statistic->statistic_quartile->outlier_low_count )
{
    list_set(
        cell_list,
        HTML_CELL *html_cell_new(
            strdup(
                string_commas_integer(
                    statistic->
                        statistic_quartile->
                        outlier_low_count ) ),
            0 /* not large_boolean */,
            0 /* not bold_boolean */ ) );
}

if (    outlier_boolean
&&  statistic->statistic_quartile->outlier_high_count )
{
    list_set(
        cell_list,
        HTML_CELL *html_cell_new(
            strdup(
                string_commas_integer(
                    statistic->
                        statistic_quartile->
                        outlier_high_count ) ),
            0 /* not large_boolean */,
            0 /* not bold_boolean */ ) );
}
```

STATISTIC (3)

```
/* statistic_table_cell_list() continued */
list_set(
    cell_list,
    HTML_CELL *html_cell_new(
        strdup(
            string_commas_money(
                statistic->
                    statistic_calculate->
                    standard_deviation ) ),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    HTML_CELL *html_cell_new(
        statistic_calculate_percent_missing_string(
            statistic->
                statistic_calculate->
                percent_missing ),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

/* Usage */
LIST *statistic_table_column_list(
    char *outlier_low_heading_string,
    char *outlier_high_heading_string );

/* Process */
LIST *list = list_new();

list_set(
    list,
    HTML_COLUMN *html_column_new(
        “Attribute”,
        0 /* not right_justify_boolean */ ) );

list_set(
    list,
    HTML_COLUMN *html_column_new(
        “Count”,
        1 /* right_justify_boolean */ ) );

list_set(
    list,
    HTML_COLUMN *html_column_new(
        “Sum”,
        1 /* right_justify_boolean */ ) );

list_set(
    list,
    HTML_COLUMN *html_column_new(
        “Minimum”,
        1 /* right_justify_boolean */ ) );

list_set(
    list,
    HTML_COLUMN *html_column_new(
        “1st Quartile”,
        1 /* right_justify_boolean */ ) );

list_set(
    list,
    HTML_COLUMN *html_column_new(
        “Average”,
        1 /* right_justify_boolean */ ) );

list_set(
    list,
    HTML_COLUMN *html_column_new(
        “Median”,
        1 /* right_justify_boolean */ ) );

list_set(
    list,
    HTML_COLUMN *html_column_new(
        “3rd Quartile”,
        1 /* right_justify_boolean */ ) );

list_set(
    list,
    HTML_COLUMN *html_column_new(
        “Maximum”,
        1 /* right_justify_boolean */ ) );

list_set(
    list,
    HTML_COLUMN *html_column_new(
        “Range”,
        1 /* right_justify_boolean */ ) );

list_set(
    list,
    HTML_COLUMN *html_column_new(
        “IQR:<br>Q3 - Q1”,
        1 /* right_justify_boolean */ ) );

list_set(
    list,
    HTML_COLUMN *html_column_new(
        “Quartile Dispersion:<br>IQR/<br>(Q3 + Q1)”,
        1 /* right_justify_boolean */ ) );

if ( outlier_low_heading_string )
{
    list_set(
        list,
        HTML_COLUMN *html_column_new(
            outlier_low_heading_string,
            1 /* right_justify_boolean */ ) );
}

if ( outlier_high_heading_string )
{
    list_set(
        list,
        HTML_COLUMN *html_column_new(
            outlier_high_heading_string,
            1 /* right_justify_boolean */ ) );
}

list_set(
    list,
    HTML_COLUMN *html_column_new(
        “Standard Deviation”,
        1 /* right_justify_boolean */ ) );

list_set(
    list,
    HTML_COLUMN *html_column_new(
        “Percent Missing”,
        1 /* right_justify_boolean */ ) );
```

STATISTIC_NUMBER_ARRAY

```
/* Usage */
STATISTIC_NUMBER_ARRAY *
    statistic_number_array_calloc(
        void );

/* Process */

/* Usage */
void statistic_number_array_insert(
    STATISTIC_NUMBER_ARRAY *
        statistic_number_array /* in/out */,
    unsigned long
        STATISTIC_NUMBER_ARRAY_MAX_SIZE,
    double statistic_value(),
    /* At least one */
    int statistic_weight() );

/* Process */
while ( statistic_weight-- )
{
    if (    statistic_number_array->array_count ==
            STATISTIC_NUMBER_ARRAY_MAX_SIZE )
    {
        max_numbers_exceeded = 1;
    }
    else
    {
        statistic_number_array->number_array[
            statistic_number_array->array_count++ ] =
                statistic_value;
    }
}

/* Usage */
void statistic_number_array_sort(
    STATISTIC_NUMBER_ARRAY *
        statistic_number_array /* in/out */ );

/* Process */
if (    !statistic_number_array
||      statistic_number_array->
        max_numbers_exceeded )
{
    return;
}

if ( statistic_number_array->array_count > 1 )
{
    void qsort(
        statistic_number_array->number_array,
        statistic_number_array->array_count,
        sizeof( double ),
        statistic_number_array_compare );
}

statistic_number_array->sorted_boolean = 1;

/* Usage */
int statistic_number_array_compare(
    const void *this,
    const void *that );

/* Process */

/* Usage */
void statistic_number_array_free(
    STATISTIC_NUMBER_ARRAY *
        statistic_number_array );

/* Process */

/* Attributes */
double number_array[
    STATISTIC_NUMBER_ARRAY_MAX_SIZE ];
unsigned long array_count;
boolean max_numbers_exceeded;
boolean sorted_boolean;
```

STATISTIC_CALCULATE

```
/* Usage */
STATISTIC_CALCULATE *statistic_calculate_new(
    STATISTIC_NUMBER_ARRAY *statistic_number_array,
    unsigned long statistic->weight_count,
    double statistic->sum,
    unsigned long statistic->row_count,
    unsigned long statistic->value_count,
    unsigned long statistic->non_zero_count );

/* Process */
STATISTIC_CALCULATE *statistic_calculate_calloc(
    void );

if ( sum )
{
    double statistic_calculate_average(
        unsigned long weight_count,
        double sum );
}

if ( statistic->row_count )
{
    double statistic_calculate_percent_missing(
        unsigned long row_count,
        unsigned long value_count );

    double statistic_calculate_non_zero_percent(
        unsigned long row_count,
        unsigned long non_zero_count );
}

if ( !statistic_number_array ) return this;

double statistic_calculate_sum_difference_squared(
    statistic_number_array,
    statistic_calculate_average() );

double statistic_calculate_variance(
    statistic_number_array->array_count,
    statistic_number_array->max_numbers_exceeded,
    statistic_calculate_sum_difference_squared() );

double statistic_calculate_standard_deviation(
    statistic_calculate_variance() );

/* Usage */
double statistic_calculate_sum_difference_squared(
    STATISTIC_NUMBER_ARRAY *statistic_number_array,
    double statistic_average() );

/* Process */
if (    !statistic_number_array
||      statistic_number_array->array_count < 2
||      statistic_number_array->
            max_numbers_exceeded )
{
    return 0.0;
}

for double number in
    statistic_number_array->number_array[]
{
    double difference = number - statistic_average;

    double sum_difference_squared +=
        ( difference * difference );
}

/* Usage */
double statistic_calculate_variance(
    int array_count,
    boolean max_numbers_exceeded,
    double
        statistic_calculate_sum_difference_squared() );

/* Process */
if (    statistic_number_array->array_count < 2
||      max_numbers_exceeded )
{
    return 0.0;
}

return
    sum_difference_squared /
    (double)( statistic_number_array->array_count - 1 );

/* Usage */
double statistic_calculate_standard_deviation(
    double statistic_calculate_variance() );

/* Process */
return sqrt( variance );

/* Usage */
char *statistic_calculate_range_string(
    double range );

/* Process */

/* Usage */
char *statistic_calculate_percent_missing_string(
    double percent_missing );

/* Process */

/* Public */
void statistic_calculate_free(
    STATISTIC_CALCULATE *statistic_calculate );

/* Attributes */
double average;
double percent_missing;
double non_zero_percent;
double sum_difference_squared;
double variance;
double standard_deviation;
```

STATISTIC_QUARTILE (1)

```
/* Usage */
STATISTIC_QUARTILE *statistic_quartile_new(
    STATISTIC_NUMBER_ARRAY *
        statistic_number_array );

/* Process */
STATISTIC_QUARTILE *statistic_quartile_calloc(
    void );

if (    !statistic_number_array
||      !statistic_number_array->array_count
||      !statistic_number_array->sorted_boolean
||      statistic_number_array->
            max_numbers_exceeded )
{
    return this;
}

double statistic_quartile_minimum(
    statistic_number_array );

double statistic_quartile_maximum(
    statistic_number_array );

double statistic_quartile_range(
    statistic_quartile_minimum(),
    statistic_quartile_maximum() );

boolean statistic_quartile_array_even_boolean(
    unsigned long statistic_number_array->
        array_count );

int statistic_quartile_median_position(
    statistic_number_array->array_count,
    statistic_quartile_array_even_boolean() );

double statistic_quartile_median(
    statistic_number_array,
    statistic_quartile_array_even_boolean(),
    statistic_quartile_median_position() );

int statistic_quartile_first_position(
    statistic_number_array->array_count,
    statistic_quartile_median_position() );

double statistic_quartile_first(
    statistic_number_array,
    statistic_quartile_array_even_boolean(),
    statistic_quartile_range(),
    statistic_quartile_first_position() );

int statistic_quartile_third_position(
    statistic_number_array->array_count,
    statistic_quartile_median_position(),
    statistic_quartile_first_position() );

double statistic_quartile_third(
    statistic_number_array,
    statistic_quartile_array_even_boolean(),
    statistic_quartile_range(),
    statistic_quartile_third_position() );

double statistic_innerquartile_range(
    double statistic_quartile_first(),
    double statistic_quartile_third() );

double statistic_quartile_coefficient_of_dispersion(
    double statistic_quartile_first(),
    double statistic_quartile_third(),
    double statistic_innerquartile_range() );

double statistic_quartile_outlier_low_limit(
    statistic_quartile_first(),
    statistic_quartile_innerquartile_range() );

double statistic_quartile_outlier_high_limit(
    statistic_quartile_third(),
    statistic_quartile_innerquartile_range() );

int statistic_quartile_outlier_low_count(
    STATISTIC_NUMBER_ARRAY *
        statistic_number_array,
    double statistic_quartile_outlier_low_limit() );

int statistic_quartile_outlier_high_count(
    STATISTIC_NUMBER_ARRAY *
        statistic_number_array,
    double statistic_quartile_outlier_high_limit() );

if ( statistic_quartile_outlier_low_count() )
{
    char *statistic_quartile_outlier_low_heading_string(
        statistic_quartile_outlier_low_limit() );
}

if ( statistic_quartile_outlier_high_count() )
{
    char *statistic_quartile_outlier_high_heading_string(
        statistic_quartile_outlier_high_limit() );
}

/* Attributes */
double minimum;
double maximum;
double range;
boolean array_even_boolean;
int median_position;
double median;
int first_position;
double first;
int third_position;
double third;
double innerquartile_range;
double coefficient_of_dispersion;
double outlier_low_limit;
double outlier_high_limit;
int outlier_low_count;
int outlier_high_count;
char *outlier_low_heading_string;
char *outlier_high_heading_string;
```

STATISTIC_QUARTILE (2)

```
/* Usage */
double statistic_quartile_minimum(
    STATISTIC_NUMBER_ARRAY *
        statistic_number_array );

/* Process */
return
    statistic_number_array->
        number_array[ 0 ];

/* Usage */
double statistic_quartile_median(
    STATISTIC_NUMBER_ARRAY *
        statistic_number_array,
    boolean statistic_quartile_array_even_boolean(),
    int statistic_quartile_median_position() );

/* Process */
if ( statistic_number_array->array_count == 1 )
{
    double median =
        statistic_number_array->number_array[ 0 ];
}
else
if ( !statistic_quartile_array_even_boolean )
{
    double median =
        statistic_number_array->
            number_array[
                statistic_quartile_median_position ];
}
else
{
    double left_value =
        statistic_number_array->
            number_array[
                statistic_quartile_median_position ];

    double right_value =
        statistic_number_array->
            number_array[
                statistic_quartile_median_position + 1 ];

    double median =
        ( left_value + right_value ) / 2.0;
}

/* Usage */
double statistic_quartile_maximum(
    STATISTIC_NUMBER_ARRAY *
        statistic_number_array );

/* Process */
return
    statistic_number_array->number_array[
        statistic_number_array->array_count - 1 ];

/* Usage */
double statistic_quartile_range(
    double statistic_quartile_minimum(),
    double statistic_quartile_maximum() );

/* Process */

/* Usage */
double statistic_quartile_first(
    STATISTIC_NUMBER_ARRAY *
        statistic_number_array,
    boolean statistic_quartile_array_even_boolean(),
    double statistic_quartile_range(),
    int statistic_quartile_first_quartile_position() );

/* Process */
if ( statistic_number_array->array_count == 1 )
{
    double first =
        statistic_number_array->
            number_array[ 0 ];
}
else
if ( statistic_number_array->array_count == 2 )
{
    double first =
        statistic_number_array->number_array[ 0 ] +
        ( statistic_quartile_range * 0.25 );
}
else
if ( !statistic_quartile_array_even_boolean )
{
    double first =
        statistic_number_array->
            number_array[
                statistic_quartile_first_position ];
}
else
{
    double left_value =
        statistic_number_array->
            number_array[
                statistic_quartile_first_position ];

    double right_value =
        statistic_number_array->
            number_array[
                statistic_quartile_first_position + 1 ];

    double first =
        ( left_value + right_value ) / 2.0;
}
```

STATISTIC_QUARTILE (3)

```
/* Usage */
double statistic_quartile_third(
    STATISTIC_NUMBER_ARRAY *
        statistic_number_array,
    boolean statistic_quartile_array_even_boolean(),
    double statistic_quartile_range(),
    int statistic_quartile_third_position() );

/* Process */
if ( statistic_number_array->array_count == 1 )
{
    double third =
        statistic_number_array->
            number_array[ 0 ];
}
else
if ( statistic_number_array->array_count == 2 )
{
    double third =
        statistic_number_array->number_array[ 1 ] -
        ( statistic_quartile_range * 0.25 );
}
else
if ( !statistic_quartile_array_even_boolean )
{
    double third =
        statistic_number_array->
            number_array[
                statistic_quartile_third_position ];
}
else
{
    double left_value =
        statistic_number_array->
            number_array[
                statistic_quartile_third_position ];

    double right_value =
        statistic_number_array->
            number_array[
                statistic_quartile_third_position + 1 ];

    double third =
        ( left_value + right_value ) / 2.0;
}

/* Usage */
double statistic_quartile_outlier_low_limit(
    double statistic_quartile_first(),
    double statistic_quartile_innerquartile_range() );

/* Process */
return
statistic_quartile_first -
1.5 X statistic_quartile_innerquartile_range;

/* Usage */
double statistic_quartile_outlier_high_limit(
    double statistic_quartile_third(),
    double statistic_quartile_innerquartile_range() );

/* Process */
return
statistic_quartile_third +
1.5 X statistic_quartile_innerquartile_range;
```

STATISTIC_QUARTILE (4)

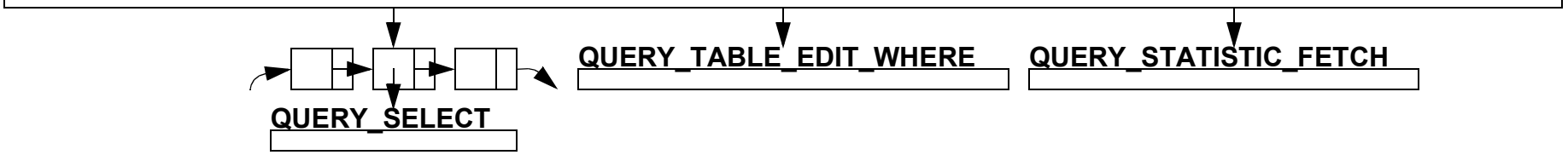
```
/* Usage */
int statistic_quartile_median_position(
    unsigned long array_count,
    boolean statistic_quartile_array_even_boolean() );

/* Process */
if ( array_count <= 2 )
{
    int median_position = 0;
}
else
if ( !statistic_quartile_array_even_boolean )
{
    int median_position =
        ( array_count - 3 ) / 2;
}
else
{
    int median_position =
        ( array_count - 2 ) / 2;
}

/* Usage */
int statistic_quartile_first_position(
    unsigned long array_count,
    int statistic_quartile_median_position() );

/* Process */
if ( array_count <= 2 )
{
    int position = 0;
}
else
/* If median position is odd */
if ( statistic_quartile_median_position % 2 )
{
    int position =
        ( statistic_quartile_median_position - 1 ) / 2;
}
else
{
    int position =
        ( statistic_quartile_median_position - 2 ) / 2;
}

/* Usage */
int statistic_quartile_third_position(
    unsigned long array_count,
    int statistic_quartile_median_position(),
    int statistic_quartile_first_position() );

/* Process */
if ( array_count <= 2 )
{
    int position = 0;
}
else
{
    int position =
        statistic_quartile_median_position +
        statistic_quartile_first_position;
}

/* Usage */
char *statistic_quartile_outlier_low_heading_string(
    double statistic_quartile_outlier_low_limit() );

/* Process */
sprintf(
    heading_string[],
    "Low Outlier Count:<br>< %s",
    string_commas_money(
        statistic_quartile_outlier_low_limit ) );

/* Usage */
char *statistic_quartile_outlier_high_heading_string(
    double statistic_quartile_outlier_high_limit() );

/* Process */
sprintf(
    heading_string[],
    "High Outlier Count:<br>> %s",
    string_commas_money(
        statistic_quartile_outlier_high_limit ) );

/* Public */
void statistic_quartile_free(
    STATISTIC_QUARTILE *statistic_quartile );
```

STATISTIC_DATE

```
/* Usage */
STATISTIC_DATE *statistic_date_new(
    int value_piece,
    char delimiter );

/* Process */
STATISTIC_DATE *statistic_date_calloc( void );

/* Usage */
void statistic_date_accumulate(
    STATISTIC_DATE *statistic_date /* in/out */,
    char *input );

/* Process */
unsigned long row_count++;

boolean statistic_empty_value(
    statistic_date->value_piece,
    statistic_date->delimiter,
    input );

if ( !statistic_empty_value() )
{
    char *statistic_date_value(
        int statistic_date->value_piece,
        char statistic_date->delimiter,
        char *input );

    unsigned long value_count++;

    void statistic_date_new_minimum_value(
        char minimum_value[] /* in/out */,
        char *statistic_date_value() );

    void statistic_date_new_maximum_value(
        char maximum_value[] /* in/out */,
        char *statistic_date_value() );

}

int range =
    date_days_between(
        minimum_value,
        maximum_value );

double percent_missing =
    statistic_calculate_percent_missing(
        row_count,
        value_count );

/* Attributes */
int value_piece;
char delimiter;
unsigned long row_count;
unsigned long value_count;
char minimum_value[];
char maximum_value[];
int range;
double percent_missing;
```

QUERY_STATISTIC (1)

```
/* Usage */
QUERY_STATISTIC *query_statistic_new(
    char *application_name,
    char *folder_name,
    /* Static memory or null */
    char *security_entity_where,
    DICTIONARY *dictionary_separate->
        query_dictionary,
    LIST *folder_attribute_append_isa_list,
    LIST *relation_mto1_list,
    LIST *relation_mto1_isa_list,
    char *primary_key_first,
    LIST *attribute_number_name_list,
    LIST *attribute_date_name_list );

/* Process */
QUERY_STATISTIC *query_statistic_calloc(
    void );

LIST *query_statistic_select_list(
    application_name,
    folder_name,
    folder_attribute_append_isa_list,
    list_length( relation_mto1_isa_list )
        /* relation_mto1_isa_list_length */,
    primary_key_first,
    attribute_number_name_list,
    attribute_date_name_list );

char *query_select_list_string(
    query_statistic_select_list(),
    (LIST *)0 /* common_name_list */ );

LIST *query_select_name_list(
    query_statistic_select_list() );

char *query_from_string(
    folder_name,
    relation_mto1_isa_list,
    (char *)0 /* row_security_role_update_list_from */ );

QUERY_TABLE_EDIT_WHERE *
    query_table_edit_where =
        query_table_edit_where_new(
            application_name,
            folder_name,
            relation_mto1_list,
            relation_mto1_isa_list,
            folder_attribute_append_isa_list,
            security_entity_where,
            query_dictionary,
            (char *)0 /* attribute_not_null */,
            (char *)0 /* join_where */ );

char *query_system_string(
    application_name,
    query_select_list_string(),
    query_from_string(),
    query_table_edit_where->string,
    (char *)0 /* group_by_string */,
    (char *)0 /* order_by_string */,
    0 /* max_rows */ );

/* Attributes */
LIST *select_list;
char *query_select_list_string;
LIST *query_select_name_list;
char *query_from_string;
QUERY_TABLE_EDIT_WHERE *
    query_table_edit_where;
char *query_system_string;

/* Set externally */
QUERY_STATISTIC_FETCH *query_statistic_fetch;
```

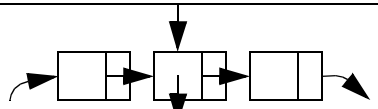


QUERY_STATISTIC (2)

```
/* Usage */
LIST *query_statistic_select_list(
    char *application_name,
    char *folder_name,
    LIST *folder_attribute_append_isa_list,
    int relation_mto1_isa_list_length,
    char *primary_key_first,
    LIST *attribute_number_name_list,
    LIST *attribute_date_name_list );

/* Process */
LIST *select_list = list_new();

list_set(
    select_list,
    QUERY_SELECT *query_select_new(
        relation_mto1_isa_list_length,
        (char *)0 /* attribute_not_null */,
        folder_name,
        primary_key_first /* attribute_name */ ) );

for char *attribute_name in
    attribute_number_name_list
{
    char *query_statistic_select_folder_name(
        folder_name,
        folder_attribute_append_isa_list,
        relation_mto1_isa_list_length,
        attribute_name );

    list_set(
        select_list,
        QUERY_SELECT *query_select_new(
            relation_mto1_isa_list_length,
            (char *)0 /* attribute_not_null */,
            query_statistic_select_folder_name(),
            attribute_name ) );
}

for char *attribute_name in attribute_date_name_list
{
    char *query_statistic_select_folder_name(
        folder_name,
        folder_attribute_append_isa_list,
        relation_mto1_isa_list_length,
        attribute_name );

    list_set(
        select_list,
        QUERY_SELECT *query_select_new(
            relation_mto1_isa_list_length,
            (char *)0 /* attribute_not_null */,
            query_statistic_select_folder_name(),
            attribute_name ) );
}

/* Usage */
char *query_statistic_select_folder_name(
    char *folder_name,
    LIST *folder_attribute_append_isa_list,
    int relation_mto1_isa_list_length,
    char *attribute_name );

/* Process */
if ( !relation_mto1_isa_list_length ) return folder_name;

FOLDER_ATTRIBUTE *folder_attribute_seek(
    (char *)0 /* folder_name */,
    attribute_name,
    folder_attribute_append_isa_list );

return
folder_attribute_seek()->folder_name;
```

QUERY_STATISTIC_FETCH

```
/* Usage */
QUERY_STATISTIC_FETCH *
   query_statistic_fetch_new(
      char SQL_DELIMITER,
      char *primary_key_first,
      LIST *attribute_number_name_list,
      LIST *attribute_date_name_list );
      char *query_system_string );

/* Process */
QUERY_STATISTIC_FETCH *
   query_statistic_fetch_calloc(
      void );

LIST *query_statistic_fetch_attribute_list(
   SQL_DELIMITER,
   primary_key_first,
   attribute_number_name_list,
   attribute_date_name_list );

FILE *query_statistic_fetch_input_pipe(
   char *query_system_string );

for char input[] in
   string_input( query_statistic_fetch_input_pipe )
{
   for QUERY_STATISTIC_FETCH_ATTRIBUTE *
      query_statistic_fetch_attribute in
         query_statistic_fetch_attribute_list
   {
      void query_statistic_fetch_attribute_accumulate(
         query_statistic_fetch_attribute,
         input );
   }
}

pclose( query_statistic_fetch_input_pipe() );

void query_statistic_fetch_attribute_calculate_set(
   attribute_list /* in/out */ );

int query_statistic_fetch_numeric_attribute_count(
   int list_length( attribute_number_name_list )
      /* attribute_number_name_list_length */ );

if (   query_statistic_fetch_numeric_attribute_count()
       == 1 )
{
   char *
   query_statistic_fetch_outlier_low_heading_string(
      LIST *attribute_list );

   char *
   query_statistic_fetch_outlier_high_heading_string(
      LIST *attribute_list );
}

/* Usage */
LIST *query_statistic_fetch_attribute_list(
   char SQL_DELIMITER,
   char *primary_key_first,
   LIST *attribute_number_name_list,
   LIST *attribute_date_name_list );

/* Process */
LIST *attribute_list = list_new();
int value_piece = 0;

boolean query_statistic_fetch_first_number_boolean(
   char *primary_key_first,
   LIST *attribute_number_name_list );

boolean query_statistic_fetch_first_date_boolean(
   char *primary_key_first,
   LIST *attribute_date_name_list );

list_set(
   attribute_list,
   query_statistic_fetch_attribute_new(
      SQL_DELIMITER,
      primary_key_first /* attribute_name */,
      value_piece++,
      1 /* attribute_is_primary_first */,
      query_statistic_fetch_first_number_boolean()
         /* attribute_is_number */,
      query_statistic_fetch_first_date_boolean()
         /* attribute_is_date */ ) );

for char *attribute_name in
   attribute_number_name_list
{
   int strcmp( attribute_name, primary_key_first );

   if ( strcmp() == 0 ) continue;

   list_set(
      attribute_list,
      query_statistic_fetch_attribute_new(
         SQL_DELIMITER,
         attribute_name,
         value_piece++,
         0 /* not attribute_is_primary_first */,
         1 /* attribute_is_number */,
         0 /* not attribute_is_date */ ) );
}

for char *attribute_name in attribute_date_name_list
{
   int strcmp( attribute_name, primary_key_first );

   if ( strcmp() == 0 ) continue;

   list_set(
      attribute_list,
      query_statistic_fetch_attribute_new(
         SQL_DELIMITER,
         attribute_name,
         value_piece++,
         0 /* not attribute_is_primary_first */,
         0 /* not attribute_is_number */,
         1 /* attribute_is_date */ ) );
}

/* Attributes */
LIST *attribute_list;
int numeric_attribute_count;
char *outlier_low_heading_string;
char *outlier_high_heading_string;
```



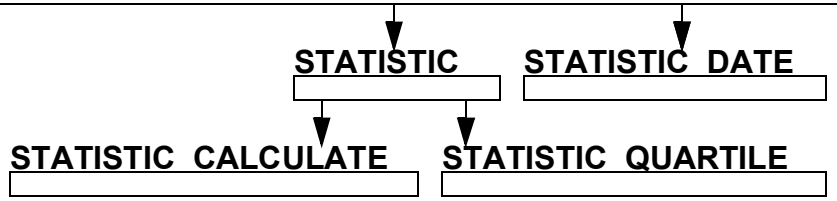
QUERY_STATISTIC_FETCH_ATTRIBUTE

QUERY_STATISTIC_FETCH_ATTRIBUTE

```
/* Usage */
QUERY_STATISTIC_FETCH_ATTRIBUTE *
    query_statistic_fetch_attribute_new(
        char SQL_DELIMITER,
        char *attribute_name,
        int value_piece,
        boolean attribute_is_primary_first,
        boolean attribute_is_number,
        boolean attribute_is_date );

/* Process */
QUERY_STATISTIC_FETCH_ATTRIBUTE *
    query_statistic_fetch_attribute_calloc(
        void );

if ( attribute_is_number )
{
    STATISTIC *statistic =
        statistic_new(
            value_piece,
            -1 /* weight_piece */,
            SQL_DELIMITER );
}
else
if ( attribute_is_date )
{
    STATISTIC_DATE *statistic_date =
        statistic_date_new(
            value_piece,
            SQL_DELIMITER );
}
```

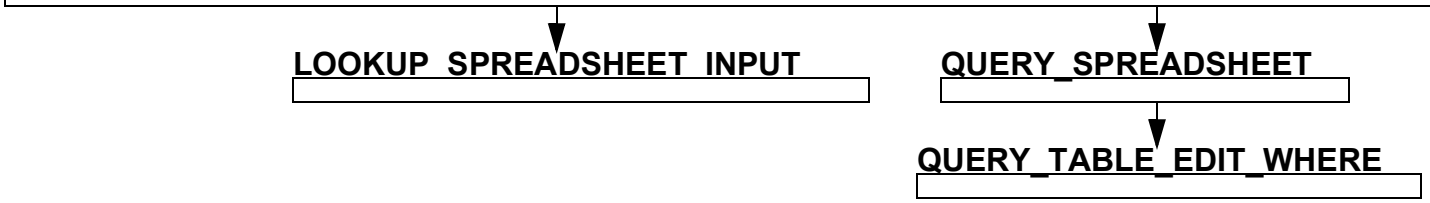
```
/* Usage */
void query_statistic_fetch_attribute_accumulate(
    QUERY_STATISTIC_FETCH_ATTRIBUTE *
        query_statistic_fetch_attribute,
    char *input );

/* Process */
if ( this->attribute_is_number )
{
    statistic_accumulate(
        this->statistic /* in/out */,
        this->statistic->statistic_number_array /* in/out */,
        input );
}

if ( attribute_is_date )
{
    statistic_date_accumulate(
        this->statistic_date /* in/out */,
        input );
}

unsigned long this->row_count++;

/* Usage */
void query_statistic_fetch_attribute_calculate_set(
    attribute_list /* in/out */ );

/* Process */
for QUERY_STATISTIC_FETCH_ATTRIBUTE *attribute
    in attribute_list
{
```

```
    if ( attribute->statistic )
    {
        attribute->statistic->statistic_calculate =
            statistic_calculate_new(
                attribute->
                    statistic->
                    statistic_number_array,
                attribute->statistic->weight_count,
                attribute->statistic->sum,
                attribute->statistic->row_count,
                attribute->statistic->value_count,
                attribute->statistic->non_zero_count );

        statistic_number_array_sort(
            attribute->
                statistic->
                statistic_number_array /* in/out */ );

        attribute->statistic->statistic_quartile =
            statistic_quartile_new(
                attribute->statistic->statistic_number_array );
    }
}

/* Attributes */
char *attribute_name;
boolean attribute_is_primary_first;
boolean attribute_is_number;
boolean attribute_is_date;
unsigned long row_count;
STATISTIC *statistic;
STATISTIC_DATE *statistic_date;
```



LOOKUP_SPREADSHEET (1)

```
/* Usage */
LOOKUP_SPREADSHEET *lookup_spreadsheet_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *dictionary_string,
    char *appaserver_parameter->data_directory );

/* Process */
LOOKUP_SPREADSHEET *
     lookup_spreadsheet_calloc(
          void );

LOOKUP_SPREADSHEET_INPUT *
    lookup_spreadsheet_input =
       lookup_spreadsheet_input_new(
          application_name,
          session_key,
          login_name,
          role_name,
          folder_name,
          dictionary_string,
          data_directory );

QUERY_SPREADSHEET *query_spreadsheet =
   query_spreadsheet_new(
      application_name,
      login_name,
      folder_name,
      lookup_spreadsheet_input->
         security_entity_where,
      lookup_spreadsheet_input->
         dictionary_separate->
         non_prefixed_dictionary
            /* query_dictionary */,
      lookup_spreadsheet_input->
         dictionary_separate->
         no_display_name_list,
      lookup_spreadsheet_input->
         role_attribute_exclude_lookup_name_list,
         lookup_spreadsheet_input->
            folder_attribute_append_isa_list,
         lookup_spreadsheet_input->
            relation_mto1_list,
         lookup_spreadsheet_input->
            relation_mto1_isa_list  );

char *lookup_spreadsheet_title_string(
    char *folder_name );

char *lookup_spreadsheet_title_string(
    char *query_spreadsheet->
       query_table_edit_where->
       string );

/* Attributes */
LOOKUP_SPREADSHEET_INPUT *
    lookup_spreadsheet_input;
QUERY_SPREADSHEET *query_spreadsheet;
char *title_string;
char *sub_title_string;
```

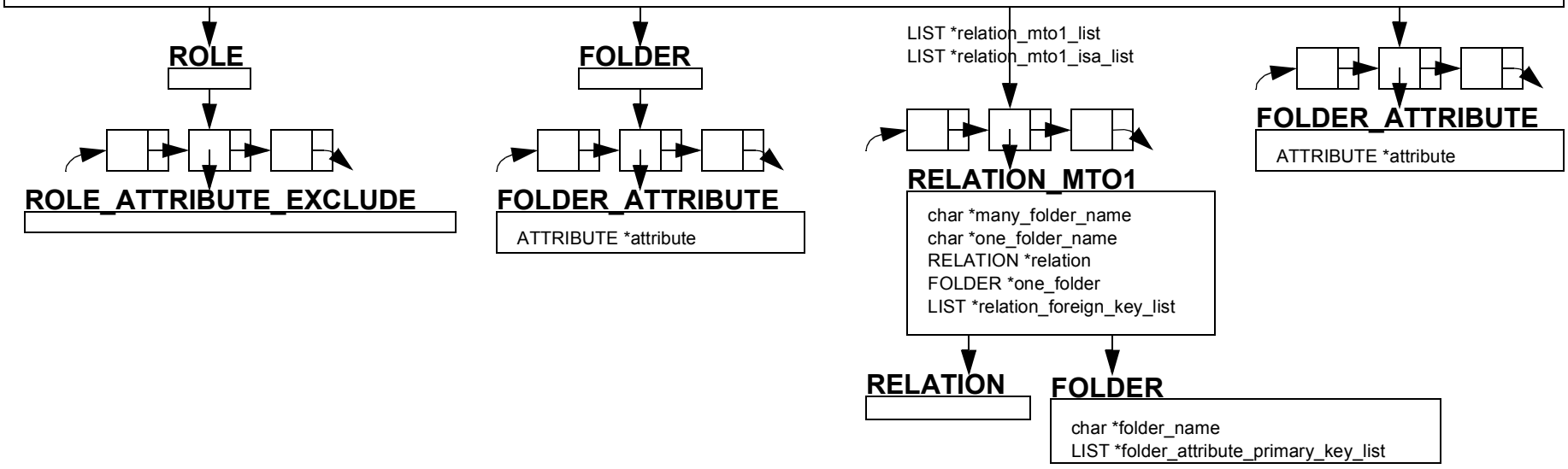


LOOKUP_SPREADSHEET (2)

```
/* Driver */
void lookup_spreadsheet_output(
    char *application_name,
    LOOKUP_SPREADSHEET *lookup_spreadsheet );

/* Process */
void document_process_output(
    application_name,
    (LIST *)0 /* javascript_filename_list */,
    lookup_spreadsheet->title_string );

printf(   "<h2>%s</h2>\n",
          lookup_spreadsheet->sub_title_string );

unsigned long row_count =
     unsigned long query_spreadsheet_output(
lookup_spreadsheet->
       lookup_spreadsheet_input->
       folder_attribute_append_isa_list,
lookup_spreadsheet->
       query_spreadsheet->
       destination_enum,
lookup_spreadsheet->
       query_spreadsheet->
       query_select_name_list,
lookup_spreadsheet->
       query_spreadsheet->
       query_system_string,
lookup_spreadsheet->
       lookup_spreadsheet_input->
       output_filename );

char *lookup_spreadsheet_prompt_message(
   unsigned long row_count );

char *appaserver_link_anchor_html(
     lookup_spreadsheet->
          lookup_spreadsheet_input->
          prompt_filename,
     "_new" /* target_window */,
     lookup_spreadsheet_prompt_message() );

printf(   "%s\n%s\n%s\n",
          appaserver_link_anchor_html(),
          document_body_close_tag(),
          document_close_tag() );
```

LOOKUP_SPREADSHEET_INPUT (1)

```
/* Usage */
LOOKUP_SPREADSHEET_INPUT *
   lookup_spreadsheet_input_new(
      char *application_name,
      char *session_key,
      char *login_name,
      char *role_name,
      char *folder_name,
      char *dictionary_string,
      char *data_directory );

/* Process */
LOOKUP_SPREADSHEET_INPUT *
   lookup_spreadsheet_input_calloc(
      void );

ROLE *role =
   role_fetch(
      role_name,
      1 /* fetch_role_attribute_exclude_list */ );

LIST *role_attribute_exclude_lookup_name_list(
   APPASERVER_LOOKUP_STATE,
   APPASERVER_UPDATE_STATE,
   role->role_attribute_exclude_list );

FOLDER *folder =
   folder_fetch(
      folder_name,
      role_attribute_exclude_lookup_name_list(),
      /* Sets folder_attribute_primary_list */
      /* Sets folder_attribute_primary_key_list */
      /* Sets folder_attribute_name_list */
      1 /* fetch_folder_attribute_list */,
      1 /* fetch_attribute */,
      1 /* cache_boolean */ );

LIST *relation_mto1_list(
   folder_name
      /* many_folder_name */,
   folder->folder_attribute_primary_key_list
      /* many_folder_primary_key_list */ );

LIST *relation_mto1_isa_list(
   (LIST *)0 /* mto1_isa_list Pass in null */,
   folder_name
      /* many_folder_name */,
   folder->folder_attribute_primary_key_list
      /* many_folder_primary_key_list */,
   0 /* not fetch_relation_one2m_list */,
   1 /* fetch_relation_mto1_list */ );

LIST *folder_attribute_append_isa_list(
   folder->folder_attribute_list,
   relation_mto1_isa_list() );

LIST *folder_attribute_date_name_list(
   folder_attribute_append_isa_list() );

LIST *folder_attribute_name_list(
   (char *)0 /* folder_name */,
   folder_attribute_append_isa_list() );
```



LOOKUP_SPREADSHEET_INPUT (2)

```
/* Process (continued) */
FOLDER_ROW_LEVEL_RESTRICTION *
   folder_row_level_restriction =
      folder_row_level_restriction_fetch(
         folder_name );

SECURITY_ENTITY *security_entity =
   security_entity_new(
      login_name,
      folder_name,
      folder_row_level_restriction->
         non_owner_forbid,
      role->override_row_restrictions );

POST_DICTIONARY *post_dictionary =
   post_dictionary_string_new(
      dictionary_string );

APPASERVER_LINK *appaserver_link =
   appaserver_link_new(
      application_http_prefix(
         application_ssl_support_boolean(
            application_name ) ),
      application_server_address(),
      data_directory,
      "spreadsheet" /* filename_stem */,
      application_name,
      0 /* process_id */,
      session_key,
      (char *)0 /* begin_date_string */,
      (char *)0 /* end_date_string */,
      "csv" /* extension */ );

char *output_filename =
   appaserver_link->
      appaserver_link_output->
      filename;

char *prompt_filename =
   appaserver_link->
      appaserver_link_prompt->
      filename;

DICTIONARY_SEPARATE_POST_PROMPT_LOOKUP *
   dictionary_separate =
      dictionary_separate_post_prompt_lookup_new(
         post_dictionary->original_post_dictionary,
         application_name,
         login_name,
         relation_mto1_list(),
         folder_attribute_name_list(),
         folder_attribute_date_name_list(),
         folder_attribute_append_isa_list() );

/* Attributes */
ROLE *role;
LIST *role_attribute_exclude_lookup_name_list;
FOLDER *folder;
LIST *relation_mto1_list;
LIST *relation_mto1_isa_list;
LIST *folder_attribute_append_isa_list;
LIST *folder_attribute_date_name_list;
LIST *folder_attribute_name_list;
FOLDER_ROW_LEVEL_RESTRICTION *
   folder_row_level_restriction;
SECURITY_ENTITY *security_entity;
POST_DICTIONARY *post_dictionary;
APPASERVER_LINK *appaserver_link;
char *output_filename;
char *prompt_filename;
DICTIONARY_SEPARATE_POST_PROMPT_LOOKUP *
   dictionary_separate;
```

SECURITY_ENTITY **POST_DICTIONARY** **APPASERVER_LINK**

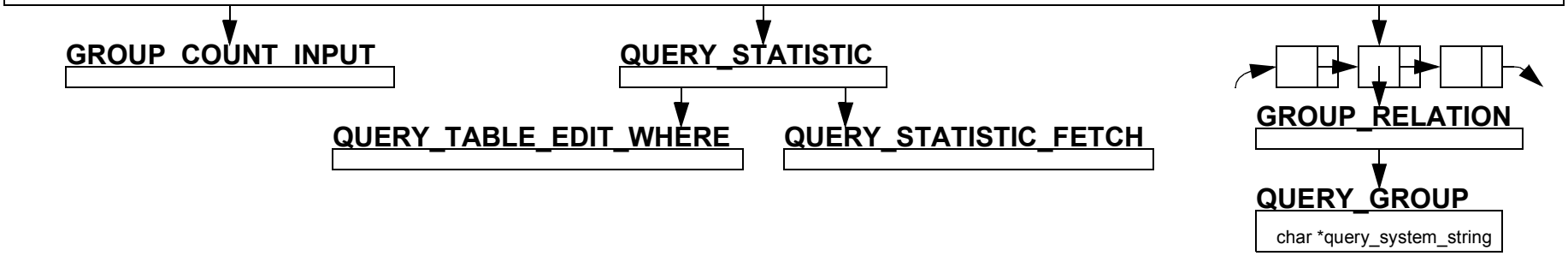
FOLDER_ROW_LEVEL_RESTRICTION

DICTIONARY_SEPARATE_POST_PROMT_LOOKUP

```
DICTIONARY *drillthru_dictionary
DICTIONARY *prompt_dictionary
DICTIONARY *no_display_dictionary
DICTIONARY *non_prefixed_dictionary
LIST *no_display_name_list
```

GROUP_COUNT (1)

```
/* Usage */
GROUP_COUNT *group_count_new(
    char *application_name,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *dictionary_string,
    char *appaserver_parameter_data_directory() );

/* Process */
GROUP_COUNT *group_count_calloc(
     void );

GROUP_COUNT_INPUT *
    group_count_input =
        group_count_input_new(
            application_name,
            login_name,
            role_name,
            folder_name,
            dictionary_string );

if ( !list_length(
        group_count_input->
            relation_mto1_list ) )
{
    exit( 1 );
}

QUERY_STATISTIC *query_statistic =
    query_statistic_new(
        application_name,
        folder_name,
        group_count_input->security_entity_where,
        group_count_input->
            dictionary_separate->
            non_prefixed_dictionary
                /* query_dictionary */,
        group_count_input->
            folder_attribute_append_isa_list,
        group_count_input->relation_mto1_list,
        group_count_input->relation_mto1_isa_list,
        group_count_input->primary_key_first,
        (LIST *)0 /* attribute_float_name_list */,
        (LIST *)0 /* attribute_date_name_list */ );

QUERY_STATISTIC_FETCH *
    query_statistic->query_statistic_fetch =
        query_statistic_fetch_new(
            SQL_DELIMITER,
            group_count_input->primary_key_first,
            (LIST *)0 /* attribute_float_name_list */,
            (LIST *)0 /* attribute_date_name_list */,
            query_statistic->query_system_string );

unsigned long group_count_row_count(
    query_statistic->query_statistic_fetch );

LIST *group_relation_list(
    application_name,
    folder_name,
    appaserver_parameter_data_directory,
    query_statistic->
        query_table_edit_where->
        string,
    group_count_input->relation_mto1_list,
    group_count_input->relation_mto1_isa_list,
    group_count_input->
        folder_attribute_float_name_list,
    group_count_input->
        dictionary_separate->
        no_display_name_list,
    group_count_row_count(),
    getpid() /* process_id */ );

char *group_count_title_string(
     char *folder_name );

char *group_count_sub_title_string(
     char *query_statistic->
          query_table_edit_where->
          string );

/* Attributes */
GROUP_COUNT_INPUT *group_count_input;
QUERY_STATISTIC *query_statistic;
unsigned long row_count;
LIST *group_relation_list;
char *title_string;
char *sub_title_string;
```



GROUP_COUNT (2)

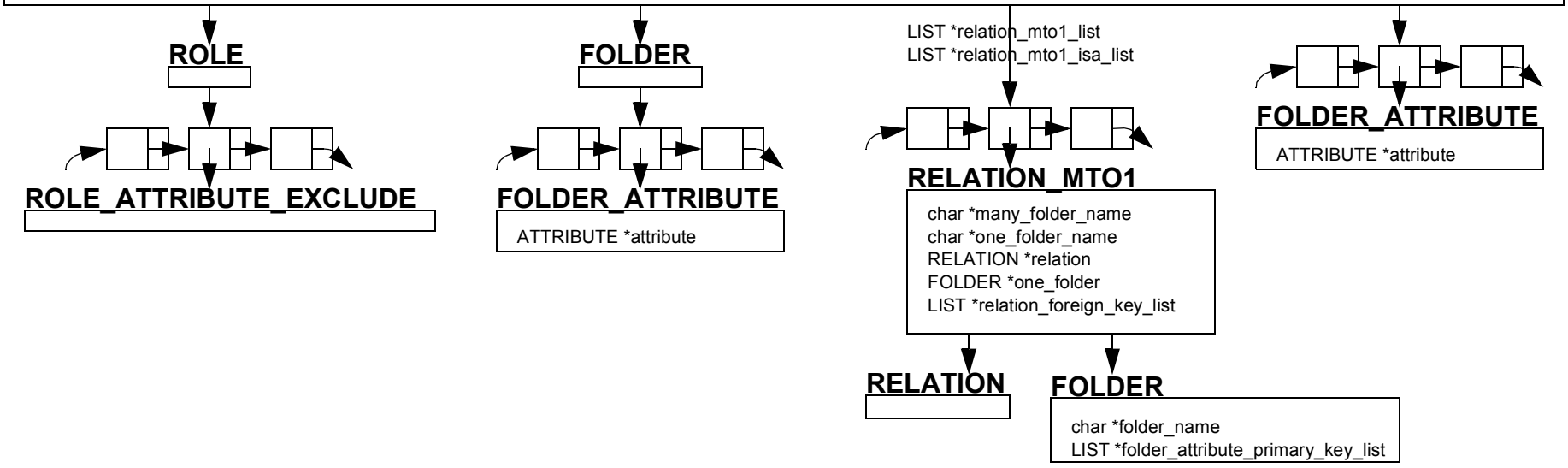
```
/* Usage */
unsigned long group_count_row_count(
    QUERY_STATISTIC_FETCH *query_statistic_fetch );

/* Process */
QUERY_STATISTIC_FETCH_ATTRIBUTE *
    query_statistic_fetch_attribute =
        list_first(
            query_statistic->
                query_statistic_fetch->
                attribute_list );

return
query_statistic_fetch_attribute->row_count;
```

```
/* Driver */
void group_count_output(
    char *application_name,
    GROUP_COUNT *group_count );

/* Process */
void document_process_output(
    application_name,
    (LIST *)0 /* javascript_filename_list */,
    group_count->title_string );

printf(   "<h2>%s</h2>\n",
          group_count->sub_title_string );
```

```
for GROUP_RELATION *group_relation in
    group_count->group_relation_list
{
    group_relation_output(
        group_count->row_count,
        group_relation );
}

printf(   "%s\n%s\n",
          document_body_close_tag(),
          document_close_tag() );
```

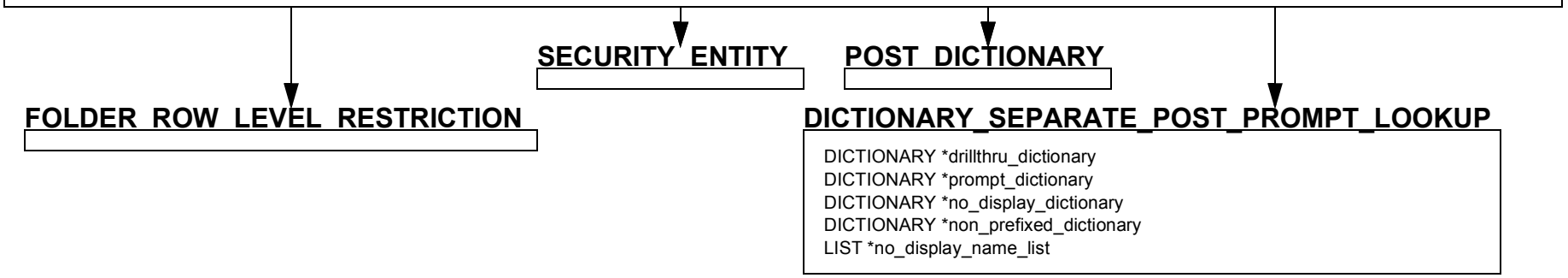
GROUP_COUNT_INPUT (1)

```
/* Usage */
GROUP_COUNT_INPUT *
   group_count_input_new(
      char *application_name,
      char *login_name,
      char *role_name,
      char *folder_name,
      char *dictionary_string );

/* Process */
GROUP_COUNT_INPUT *
   group_count_input_calloc(
      void );

ROLE *role =
   role_fetch(
      role_name,
      1 /* fetch_role_attribute_exclude_list */ );

LIST *role_attribute_exclude_lookup_name_list(
   APPASERVER_LOOKUP_STATE,
   APPASERVER_UPDATE_STATE,
   role->role_attribute_exclude_list );

FOLDER *folder =
   folder_fetch(
      folder_name,
      role_attribute_exclude_lookup_name_list(),
      /* Sets folder_attribute_primary_list */
      /* Sets folder_attribute_primary_key_list */
      /* Sets folder_attribute_name_list */
      1 /* fetch_folder_attribute_list */,
      1 /* fetch_attribute */,
      1 /* cache_boolean */ );

LIST *relation_mto1_list(
   folder_name
      /* many_folder_name */,
   folder->folder_attribute_primary_key_list
      /* many_folder_primary_key_list */ );

LIST *relation_mto1_isa_list(
   (LIST *)0 /* mto1_isa_list Pass in null */,
   folder_name
      /* many_folder_name */,
   folder->folder_attribute_primary_key_list
      /* many_folder_primary_key_list */,
   0 /* not fetch_relation_one2m_list */,
   1 /* fetch_relation_mto1_list */ );

LIST *folder_attribute_append_isa_list(
   folder->folder_attribute_list,
   relation_mto1_isa_list() );

LIST *folder_attribute_name_list(
   (char *)0 /* folder_name */,
   folder_attribute_append_isa_list() );

LIST *folder_attribute_date_name_list(
   folder_attribute_append_isa_list() );

LIST *folder_attribute_float_name_list(
   folder_attribute_append_isa_list() );
```



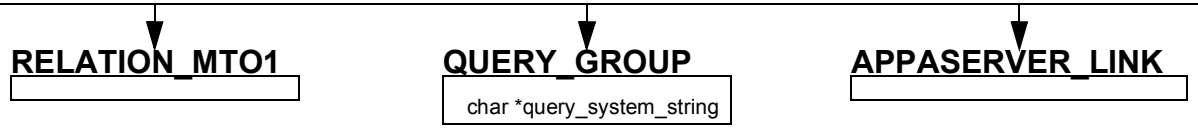
GROUP_COUNT_INPUT (2)

```
/* Process (continued) */
FOLDER_ROW_LEVEL_RESTRICTION *
    folder_row_level_restriction =
        folder_row_level_restriction_fetch(
            folder_name );

SECURITY_ENTITY *security_entity =
    security_entity_new(
        login_name,
        folder_name,
        folder_row_level_restriction->
            non_owner_forbid,
        role->override_row_restrictions );

POST_DICTIONARY *post_dictionary =
    post_dictionary_string_new(
        dictionary_string );

DICTIONARY_SEPARATE_POST_PROMPT_LOOKUP *
    dictionary_separate =
        dictionary_separate_post_prompt_lookup_new(
            post_dictionary->original_post_dictionary,
            application_name,
            login_name,
            relation_mto1_list(),
            folder_attribute_name_list(),
            folder_attribute_date_name_list(),
            folder_attribute_append_isa_list() );

char *group_count_input_primary_key_first(
    char *list_first(
        folder->
            folder_attribute_primary_key_list )
                /* primary_key_first */ );

/* Attributes */
ROLE *role;
LIST *role_attribute_exclude_lookup_name_list;
FOLDER *folder;
LIST *relation_mto1_list;
LIST *relation_mto1_isa_list;
LIST *folder_attribute_append_isa_list;
LIST *folder_attribute_date_name_list;
LIST *folder_attribute_float_name_list;
FOLDER_ROW_LEVEL_RESTRICTION *
    folder_row_level_restriction;
SECURITY_ENTITY *security_entity;
POST_DICTIONARY *post_dictionary;
DICTIONARY_SEPARATE_POST_PROMPT_LOOKUP *
    dictionary_separate;
char *primary_key_first;
```



GROUP_RELATION (1)

```
/* Usage */
LIST *group_relation_list(
    char *application_name,
    char *folder_name,
    char *data_directory,
    char *query_table_edit_where_string,
    LIST *relation_mto1_list,
    LIST *relation_mto1_isa_list,
    LIST *folder_attribute_float_name_list,
    LIST *dictionary_separate->no_display_name_list,
    unsigned long group_count_row_count,
    pid_t process_id );

/* Process */
LIST *relation_list = list_new();

for RELATION_MTO1 *relation_mto1 in
    relation_mto1_list
{
    pid_t group_relation_process_id_increment(
        pid_t process_id );

    LIST *list_subtract(
        relation_mto1->relation_foreign_key_list,
        no_display_name_list );

    boolean list_length( list_subtract );

    if ( !list_length() ) continue;

    list_set(
        relation_list,
        GROUP_RELATION *group_relation_new(
            application_name,
            folder_name,
            data_directory,
            query_table_edit_where_string,
            relation_mto1_isa_list,
            folder_attribute_float_name_list,
            group_count_row_count,
            relation_mto1,
            group_relation_process_id_increment() ) );
}

/* Usage */
GROUP_RELATION *group_relation_new(
    char *application_name,
    char *session_key,
    char *folder_name,
    char *document_root_directory,
    char *query_table_edit_where_string,
    LIST *relation_mto1_isa_list,
    LIST *folder_attribute_float_name_list,
    unsigned long group_count_row_count(),
    RELATION_MTO1 *relation_mto1,
    pid_t group_relation_process_id_increment() );

/* Process */
GROUP_RELATION *group_relation_calloc( void );

QUERY_GROUP *query_group =
    query_group_new(
        application_name,
        folder_name,
        query_table_edit_where_string,
        relation_mto1_isa_list,
        folder_attribute_float_name_list,
        group_count_row_count,
        relation_mto1->relation_foreign_key_list );

APPASERVER_LINK *appaserver_link =
    appaserver_link_new(
        application_http_prefix( application_name ),
        application_server_address(),
        (application_prepend_http_protocol_yn(
            application_name) == 'y'),
        data_directory,
        "group_relation" /* filename_stem */,
        application_name,
        group_relation_process_id_increment(),
        (char *)0 /* session_key */,
        (char *)0 /* begin_date_string */,
        (char *)0 /* end_date_string */,
        "csv" /* extension */ );

char *prompt_filename =
    appaserver_link->
        appaserver_link_prompt->
        filename;

char *output_filename =
    appaserver_link->
        appaserver_link_output->
        filename;

char *group_relation_anchor_html(
    relation_mto1->one_folder_name,
    prompt_filename );

char *group_relation_title(
    char *folder_name,
    char *relation_mto1->relation_prompt );

char *group_relation_html_table_system_string(
    SQL_DELIMITER,
    HTML_TABLE_QUEUE_TOP_BOTTOM,
    folder_attribute_float_name_list,
    relation_mto1->relation_foreign_key_list,
    group_relation_title() );

char *group_relation_csv_heading_string(
    LIST *folder_attribute_float_name_list,
    LIST *relation_mto1->relation_foreign_key_list );

char *group_relation_csv_heading_system_string(
    char *output_filename,
    char *group_relation_csv_heading_string() );

char *group_relation_csv_output_system_string(
    SQL_DELIMITER,
    output_filename );

/* Attributes */
RELATION_MTO1 *relation_mto1;
char *title;
QUERY_GROUP *query_group;
APPASERVER_LINK *appaserver_link;
char *prompt_filename;
char *output_filename;
char *anchor_html;
char *html_table_system_string;
char *csv_heading_string;
char *csv_heading_system_string;
char *csv_output_system_string;
```



GROUP_RELATION (2)

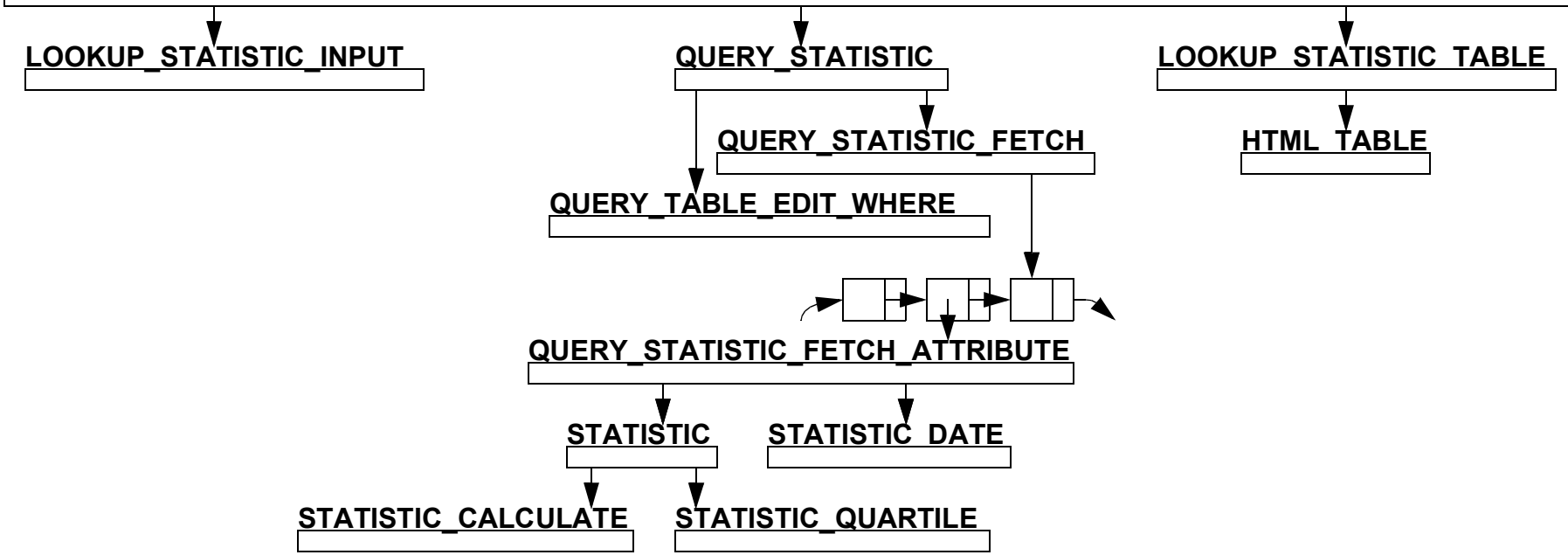
```
/* Usage */
char *group_relation_html_table_system_string(
    const char SQL_DELIMITER,
    int HTML_TABLE_QUEUE_TOP_BOTTOM,
    LIST *folder_attribute_float_name_list,
    LIST *relation_mto1->relation_foreign_key_list,
    char *group_relation_title() );

/* Process */
static char *group_relation_piece_commas_double_string(
    const char *SQL_DELIMITER,
    int list_length( folder_attribute_float_name_list )
        /* folder_attribute_float_name_list_length */ );

char *group_relation_html_heading_string(
    LIST *folder_attribute_float_name_list,
    LIST *relation_foreign_key_list );

char *group_relation_html_justify_string(
    int list_length( relation_foreign_key_list )
        /* foreign_key_list_length */ );

snprintf(
    system_string[],
    sizeof ( system_string ),
    "queue_top_bottom_lines.e %d |"
    "piece_commas_double.e 1 0 '%c' |"
    "%s |"
    "html_table.e '^^%s' '%s' '%c' '%s'",
    HTML_TABLE_QUEUE_TOP_BOTTOM,
    SQL_DELIMITER,
    group_relation_piece_commas_double_string(),
    group_relation_title,
    group_relation_html_heading_string(),
    SQL_DELIMITER,
    group_relation_html_justify_string() );

/* Usage */
char *group_relation_csv_output_system_string(
    const char SQL_DELIMITER,
    char *output_filename );

/* Process */
snprintf(
    system_string[],
    sizeof ( system_string ),
    "double_quote_comma_delimited.e '%c' >> %s",
    SQL_DELIMITER,
    output_filename );

/* Usage */
char *group_relation_anchor_html(
    char *one_folder_name,
    char *prompt_filename );

/* Process */
char *group_relation_prompt_message(
    char *one_folder_name );

char *appaserver_link_anchor_html(
     prompt_filename,
     "_new" /* target_window */,
     group_relation_prompt_message() );

/* Usage */
void group_relation_output(
    unsigned long group_count->row_count,
    GROUP_RELATION *group_relation );

/* Process */
system( group_relation->csv_heading_system_string );

FILE *group_relation_csv_output_pipe(
    char *group_relation->csv_output_system_string );

FILE *group_relation_html_output_pipe(
    char *group_relation->html_table_system_string );

FILE *group_relation_input_pipe(
    char *group_relation->
        query_group->
        query_system_string );

for char input[] in string_input(
    group_relation_input_pipe() )
{
    fprintf(
        group_relation_html_output_pipe(),
        "%s\n",
        input );

    fprintf(
        group_relation_csv_output_pipe(),
        "%s\n",
        input );
}

char *group_relation_html_row_count_string(
    char SQL_DELIMITER,
    unsigned long group_count_row_count,
    int list_length(
        group_relation->
            relation_mto1->
            relation_foreign_key_list )
                /* foreign_key_list_length */ );

fprintf(
    group_relation_html_output_pipe(),
    "%s\n",
    group_relation_html_row_count_string() );

void pclose( group_relation_input_pipe() );
void pclose( group_relation_html_output_pipe() );
void pclose( group_relation_csv_output_pipe() );

printf(
    "%s<p>\n",
    group_relation->anchor_html );

fflush( stdout );
```

LOOKUP_STATISTIC (1)

```
/* Usage */
LOOKUP_STATISTIC *lookup_statistic_new(
    char *application_name,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *dictionary_string );

/* Process */
LOOKUP_STATISTIC *lookup_statistic_calloc(
    void );

LOOKUP_STATISTIC_INPUT *
    lookup_statistic_input =
        lookup_statistic_input_new(
            application_name,
            login_name,
            role_name,
            folder_name,
            dictionary_string );

QUERY_STATISTIC *query_statistic =
    query_statistic_new(
        application_name,
        folder_name,
        lookup_statistic_input->security_entity_where,
        lookup_statistic_input->
            dictionary_separate->
            non_prefixed_dictionary
                /* query_dictionary */,
        lookup_statistic_input->
            folder_attribute_append_isa_list,
        lookup_statistic_input->relation_mto1_list,
        lookup_statistic_input->relation_mto1_isa_list,
        lookup_statistic_input->primary_key_first,
        lookup_statistic_input->
            attribute_number_name_list,
        lookup_statistic_input->
            attribute_date_name_list );

query_statistic->query_statistic_fetch =
    query_statistic_fetch_new(
        SQL_DELIMITER,
        lookup_statistic_input->primary_key_first,
        lookup_statistic_input->
            attribute_number_name_list,
        lookup_statistic_input->
            attribute_date_name_list,
        query_statistic->query_system_string );

int lookup_statistic_after_minimum_blank_count(
    void );

int lookup_statistic_after_range_blank_count(
        query_statistic->
            query_statistic_fetch->
            outlier_low_heading_string,
        query_statistic->
            query_statistic_fetch->
            outlier_high_heading_string );

LOOKUP_STATISTIC_TABLE *lookup_statistic_table =
    lookup_statistic_table_new(
        lookup_statistic_input->attribute_float_count,
        query_statistic->
                query_statistic_fetch->
                attribute_list,
        query_statistic->
            query_statistic_fetch->
            outlier_low_heading_string,
        query_statistic->
            query_statistic_fetch->
            outlier_high_heading_string,
        lookup_statistic_after_minimum_blank_count(),
        lookup_statistic_after_range_blank_count() );

char *lookup_statistic_sub_title_string(
    char *query_statistic->
        query_table_edit_where->
        string );

/* Attributes */
LOOKUP_STATISTIC_INPUT *lookup_statistic_input;
QUERY_STATISTIC *query_statistic;
int after_minimum_blank_count;
int after_range_blank_count;
LOOKUP_STATISTIC_TABLE *lookup_statistic_table;
char *title_html;
```

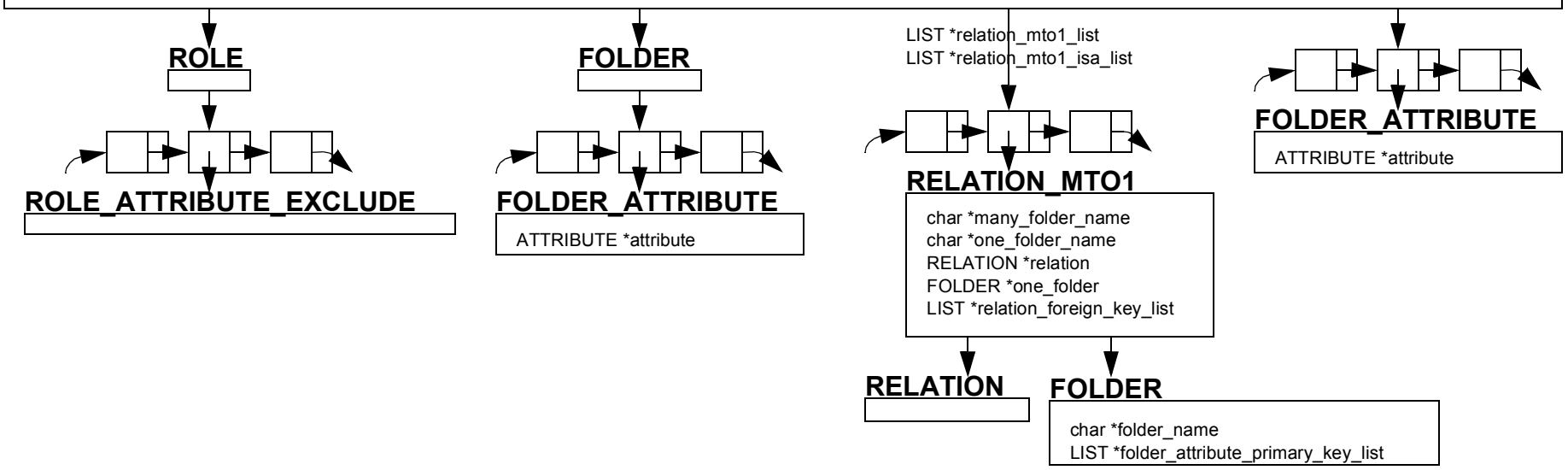
LOOKUP_STATISTIC (2)

```
/* Driver */
void lookup_statistic_output(
    char *application_name,
    LOOKUP_STATISTIC *lookup_statistic );

/* Process */
void document_process_output(
    application_name,
    (LIST *)0 /* javascript_filename_list */,
    (char *)0 /* title_string */ );

printf(   "%s\n",
          lookup_statistic->title_html );

void html_table_output(
    lookup_statistic->
          lookup_statistic_table->
          html_table,
        0 /* html_table_rows_between_heading */ );

printf(   "%s\n%s\n",
          document_body_close_tag(),
          document_close_tag() );
```



LOOKUP_STATISTIC_INPUT (1)

```
/* Usage */
LOOKUP_STATISTIC_INPUT *
   lookup_statistic_input_new(
      char *application_name,
      char *login_name,
      char *role_name,
      char *folder_name,
      char *dictionary_string );

/* Process */
LOOKUP_STATISTIC_INPUT *
   lookup_statistic_input_calloc(
      void );

ROLE *role =
   role_fetch(
      role_name,
      1 /* fetch_role_attribute_exclude_list */ );

LIST *role_attribute_exclude_lookup_name_list(
   APPASERVER_LOOKUP_STATE,
   APPASERVER_UPDATE_STATE.
   role->role_attribute_exclude_list );

FOLDER *folder =
   folder_fetch(
      folder_name,
      role_attribute_exclude_lookup_name_list(),
      /* Sets folder_attribute_primary_list */
      /* Sets folder_attribute_primary_key_list */
      /* Sets folder_attribute_name_list */
      1 /* fetch_folder_attribute_list */,
      1 /* fetch_attribute */,
      1 /* cache_boolean */ );

LIST *relation_mto1_list(
   folder_name
      /* many_folder_name */,
   folder->folder_attribute_primary_key_list
      /* many_folder_primary_key_list */ );

LIST *relation_mto1_isa_list(
   (LIST *)0 /* mto1_isa_list Pass in null */,
   folder_name
      /* many_folder_name */,
   folder->folder_attribute_primary_key_list
      /* many_folder_primary_key_list */,
   0 /* not fetch_relation_one2m_list */,
   1 /* fetch_relation_mto1_list */ );

LIST *folder_attribute_append_isa_list(
   folder->folder_attribute_list,
   relation_mto1_isa_list() );

char *lookup_statistic_input_primary_key_first(
   char *list_first(
      folder->
         folder_attribute_primary_key_list )
            /* primary_key_first */ );
```



LOOKUP_STATISTIC_INPUT (2)

```
/* Process (continued) */
FOLDER_ROW_LEVEL_RESTRICTION *
    folder_row_level_restriction =
        folder_row_level_restriction_fetch(
            folder_name );

SECURITY_ENTITY *security_entity =
    security_entity_new(
        login_name,
        folder_name,
        folder_row_level_restriction->
            non_owner_forbid,
        role->override_row_restrictions );

POST_DICTIONARY *post_dictionary =
    post_dictionary_string_new(
        dictionary_string );

LIST *folder_attribute_number_name_list(
    folder_attribute_append_isa_list() );

LIST *folder_attribute_date_name_list(
    folder_attribute_append_isa_list() );

DICTIONARY_SEPARATE_POST_PROMPT_LOOKUP *
    dictionary_separate =
        dictionary_separate_post_prompt_lookup_new(
            post_dictionary->original_post_dictionary,
            application_name,
            login_name,
            relation_mto1_list(),
            foler_attribute_name_list(),
            folder_attribute_date_name_list(),
            folder_attribute_append_isa_list() );

LIST *
    lookup_statistic_input_attribute_number_name_list(
        LIST *folder_attribute_number_name_list(),
        LIST *dictionary_separate->
            no_display_name_list );

int lookup_statistic_input_attribute_number_count(
    LIST *lookup_statistic_input_attribute_number_name_list );

LIST *lookup_statistic_input_attribute_date_name_list(
    LIST *folder_attribute_date_name_list(),
        LIST *dictionary_separate->no_display_name_list );

/* Attributes */
ROLE *role;
LIST *role_attribute_exclude_lookup_name_list;
FOLDER *folder;
LIST *relation_mto1_list;
LIST *relation_mto1_isa_list;
LIST *folder_attribute_append_isa_list;
char *primary_key_first;
FOLDER_ROW_LEVEL_RESTRICTION *
    folder_row_level_restriction;
SECURITY_ENTITY *security_entity;
POST_DICTIONARY *post_dictionary;
LIST *folder_attribute_float_name_list;
LIST *folder_attribute_date_name_list;
DICTIONARY_SEPARATE_POST_PROMPT_LOOKUP *
    dictionary_separate;
LIST *attribute_number_name_list;
int attribute_number_count;
LIST *attribute_date_name_list;
```

SECURITY_ENTITY

POST_DICTIONARY

DRILLTHRU_STATUS

FOLDER_ROW_LEVEL_RESTRICTION

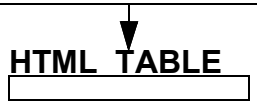
DICTIONARY_SEPARATE_POST_PROMPT_LOOKUP

```
DICTIONARY *drillthru_dictionary
DICTIONARY *prompt_dictionary
DICTIONARY *no_display_dictionary
DICTIONARY *non_prefixed_dictionary
LIST *no_display_name_list
```

LOOKUP_STATISTIC_TABLE (1)

```
/* Usage */
LOOKUP_STATISTIC_TABLE *
   lookup_statistic_table_new(
      int lookup_statistic_input->
         attribute_number_count,
      LIST *query_statistic->
            query_statistic_fetch->
            query_statistic_fetch_attribute_list,
      char *outlier_low_heading_string,
      char *outlier_high_heading_string,
      int lookup_statistic_after_minimum_blank_count(),
      int lookup_statistic_after_range_blank_count() );
```

```
/* Process */
LOOKUP_STATISTIC_TABLE *
   lookup_statistic_table_calloc(
      void );

HTML_TABLE *html_table =
   html_table_new(
      (char *)0 /* title */,
      (char *)0 /* sub_title */,
      (char *)0 /* sub_sub_title */ );

html_table->column_list =
   statistic_table_column_list(
      outlier_low_heading_string,
      outlier_high_heading_string );

html_table->row_list =
   lookup_statistic_table_row_list(
      lookup_statistic_input->attribute_number_count,
      query_statistic_fetch_attribute_list,
      lookup_statistic_after_minimum_blank_count,
      lookup_statistic_after_range_blank_count );

/* Attribute */
HTML_TABLE *html_table;
```



LOOKUP_STATISTIC_TABLE (2)

```
/* Usage */
LIST *lookup_statistic_table_row_list(
    int lookup_statistic_input->attribute_number_count,
    LIST *query_statistic_fetch_attribute_list,
    int lookup_statistic_after_minimum_blank_count(),
    int lookup_statistic_after_range_blank_count() );

/* Process */
LIST *row_list = list_new();

for QUERY_STATISTIC_FETCH_ATTRIBUTE *
    query_statistic_fetch_attribute in
        query_statistic_fetch_attribute_list
{
    list_set(
        row_list,
        HTML_ROW *lookup_statistic_table_row_new(
            lookup_statistic_input->
                attribute_number_count,
            query_statistic_fetch_attribute,
            lookup_statistic_after_minimum_blank_count,
            lookup_statistic_after_range_blank_count ) );
}

/* Usage */
HTML_ROW *lookup_statistic_table_row_new(
    int lookup_statistic_input->attribute_float_count,
    QUERY_STATISTIC_FETCH_ATTRIBUTE *
        query_statistic_fetch_attribute,
        int lookup_statistic_after_minimum_blank_count(),
        int lookup_statistic_after_range_blank_count() );

/* Process */
HTML_ROW *html_row = html_row_calloc();
static int row_number;

if ( row_number++ % 2 )
{
    html_row->background_shaded_boolean = 1;
}

html_row->cell_list =
    LIST *lookup_statistic_table_cell_list(
        lookup_statistic_input->attribute_number_count,
        query_statistic_fetch_attribute,
        lookup_statistic_after_minimum_blank_count,
        lookup_statistic_after_range_blank_count );

/* Usage */
LIST *lookup_statistic_table_cell_list(
    int lookup_statistic_input->attribute_number_count,
    QUERY_STATISTIC_FETCH_ATTRIBUTE *
        query_statistic_fetch_attribute,
    int lookup_statistic_after_minimum_blank_count,
    int lookup_statistic_after_range_blank_count );

/* Process */
if ( query_statistic_fetch_attribute->
        attribute_is_primary_first )
{
    char *lookup_statistic_table_primary_label(
        char *query_statistic_fetch_attribute->
            attribute_name );

    if ( query_statistic_fetch_attribute->
            attribute_is_float )
    {
        return
        LIST *lookup_statistic_table_number_cell_list(
            attribute_number_count,
            query_statistic_fetch_attribute->statistic,
            lookup_statistic_table_primary_label() );
    }
    else
    if ( query_statistic_fetch_attribute->
            attribute_is_date )
    {
        return
        LIST *lookup_statistic_table_date_cell_list(
            query_statistic_fetch_attribute->
                statistic_date,
            lookup_statistic_table_primary_label(),
            lookup_statistic_after_minimum_blank_count,
            lookup_statistic_after_range_blank_count );
    }
    else
    {
        return
        LIST *lookup_statistic_table_primary_cell_list(
            query_statistic_fetch_attribute->row_count,
            lookup_statistic_table_primary_label(),
            lookup_statistic_after_minimum_blank_count,
            lookup_statistic_after_range_blank_count );
    }
}

if (    query_statistic_fetch_attribute->
            attribute_is_number )
{
    char *lookup_statistic_table_label(
        query_statistic_fetch_attribute->
            attribute_name );

    return
    LIST *lookup_statistic_table_number_cell_list(
        attribute_number_count,
        query_statistic_fetch_attribute->statistic,
        lookup_statistic_table_label() );
}

if (    query_statistic_fetch_attribute->
            attribute_is_date )
{
    char *lookup_statistic_table_label(
        query_statistic_fetch_attribute->
            attribute_name );

    return
    LIST *lookup_statistic_table_date_cell_list(
        query_statistic_fetch_attribute->statistic_date,
        lookup_statistic_table_label(),
        lookup_statistic_after_minimum_blank_count,
        lookup_statistic_after_range_blank_count );
}
exit( 1 );
```

LOOKUP_STATISTIC_TABLE (3)

```
/* Usage */
LIST *lookup_statistic_table_float_cell_list(
    int lookup_statistic_input->attribute_number_count,
    STATISTIC *statistic,
    char *label );

/* Process */
boolean lookup_statistic_table_outlier_boolean(
    int attribute_number_count );

LIST *statistic_table_cell_list(
    statistic,
    label,
    lookup_statistic_table_outlier_boolean()  );

/* Usage */
LIST *lookup_statistic_table_date_cell_list(
    STATISTIC_DATE *statistic_date,
    char *label,
    int lookup_statistic_after_minimum_blank_count(),
    int lookup_statistic_after_range_blank_count() );

/* Process */
LIST *cell_list = list_new();

list_set(
      cell_list,
      HTML_CELL *html_cell_new(
            label /* data */,
            0 /* not large_boolean */,
            0 /* not bold_boolean */ ) );

char *string_commas_unsigned_long(
    statistic_date->value_count );

list_set(
    cell_list,
    HTML_CELL *html_cell_new(
        strdup(
            string_commas_unsigned_long() ),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new( (char *)0 /* sum */, 0, 0 ) );

list_set(
    cell_list,
    HTML_CELL *html_cell_new(
        statistic_date->minimum_value,
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

for (  int i = 0;
       i < lookup_statistic_after_minimum_blank_count;
       i++ )
{
       list_set(
           cell_list,
           html_cell_new( (char *)0, 0, 0 ) );
}

list_set(
    cell_list,
    HTML_CELL *html_cell_new(
        statistic_date->maximum_value,
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    HTML_CELL *html_cell_new(
        statistic_quartile_range_string(
            (double)statistic_date->range ),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

for (  int i = 0;
       i < lookup_statistic_after_range_blank_count;
       i++ )
{
       list_set(
           cell_list,
           html_cell_new( (char *)0, 0, 0 ) );
}

list_set(
    cell_list,
    HTML_CELL *html_cell_new(
        statistic_calculate_percent_missing_string(
            statistic_date->percent_missing ),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );
```

LOOKUP_STATISTIC_TABLE (4)

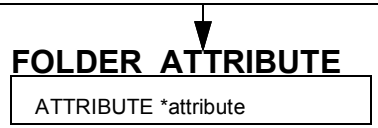
```
/* Usage */
LIST *lookup_statistic_table_primary_cell_list(
    unsigned long row_count,
    char *lookup_statistic_table_primary_label(),
    int lookup_statistic_after_minimum_blank_count(),
    int lookup_statistic_after_range_blank_count );

/* Process */
LIST *cell_list = list_new();

list_set(
    cell_list,
    HTML_CELL *html_cell_new(
        lookup_statistic_table_primary_label,
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    HTML_CELL *html_cell_new(
        place_commas_in_unsigned_long(
            row_count ),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

int lookup_statistic_table_blank_count(
    int lookup_statistic_after_minimum_blank_count,
    int lookup_statistic_after_blank_count );

for (  int i = 0;
        i < lookup_statistic_table_blank_count();
        i++ )
{
        list_set(
            cell_list,
            html_cell_new( (char *)0, 0, 0 ) );
}

list_set(
    cell_list,
    HTML_CELL *html_cell_new(
        statistic_calculate_percent_missing_string(
            0.0 /* percent_missing */ ),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

/* Usage */
char *lookup_statistic_table_label(
    char *query_statistic_fetch_attribute->
        attribute_name );

/* Process */
char *strdup(
    string_initial_capital(
        destination[],
        attribute_name ) );
```

SHELL_SCRIPT

```
/* Usage */
SHELL_SCRIPT *shell_script_new(
    char *application_name,
    char *shell_script_filespecification,
    LIST *sql_statement_list,
    boolean tee_appaserver_boolean,
    boolean html_paragraph_wrapper_boolean );

/* Process */
SHELL_SCRIPT *shell_script_calloc(
    void );

FILE *appaserver_output_file(
    shell_script_filespecification );

char *shell_script_open_string(
    char *application_name );

fprintf( appaserver_output_file(),
    "%s\n",
    shell_script_open_string() );

char *shell_script_here_open_string(
    const char *SHELL_SCRIPT_HERE_LABEL );

fprintf( appaserver_output_file(),
    "%s\n",
    shell_script_here_open_string() );

for char *sql_statement in sql_statement_list
{
    fprintf( appaserver_output_file(),
        "%s\n",
        sql_statement );
}

char *shell_script_here_close_string(
    const char *SHELL_SCRIPT_HERE_LABEL,
    boolean tee_appaserver_boolean,
    boolean html_paragraph_wrapper_boolean );

fprintf( appaserver_output_file(),
    "%s\n",
    shell_script_here_close_string() );

char *shell_script_close_string(
    void );

fprintf( appaserver_output_file(),
    "%s\n",
    char *shell_script_close_string() );

fclose( appaserver_output_file() );

char *shell_script_execute_bit_system_string(
    shell_script_filespecification );

system( shell_script_execute_bit_system_string() );

/* Usage */
char *shell_script_filespecification(
    const char *PROCESS_LABEL,
    char *application_name,
    char *folder_name,
    char *data_directory );

/* Process */

/*  Usage */
void shell_script_spool_string_list(
    char *application_name,
    char *shell_filename,
    LIST *sql_statement_list );

/* Process */

/* Usage */
char *shell_script_execute_bit_system_string(
    char *shell_script_filespecification );

/* Process */

/* Attributes */
```

ADD_COLUMN (1)

```
/* Usage */
ADD_COLUMN *add_column_new(
    char *application_name,
    char *security_sql_injection_escape(
        folder_name ),
    char *security_sql_injection_escape(
        attribute_name ),
    char *appaserver_parameter->data_directory );

/* Process */
ADD_COLUMN *add_column_calloc(
    void );

FOLDER_ATTRIBUTE *folder_attribute =
    folder_attribute_fetch(
        folder_name,
        attribute_name,
        1 /* fetch_attribute */ );

char *appaserver_table_name(
    folder_name );

char *attribute_database_datatype(
    folder_attribute->attribute->datatype_name,
    folder_attribute->attribute->width,
    folder_attribute->attribute->float_decimal_places,
    folder_attribute->primary_key_index,
    folder_attribute->default_value );

char *add_column_execute_system_string(
    attribute_name,
    appaserver_table_name(),
    attribute_database_datatype() );

char *add_column_shell_script(
    application_name,
    folder_attribute,
    appaserver_table_name,
    add_column_execute_system_string() );

char *add_column_process_filespecification(
    application_name,
    folder_name,
    attribute_name,
    data_directory );

/* Usage */
char *add_column_execute_system_string(
    char *attribute_name,
    char *appaserver_table_name(),
    char *attribute_database_datatype() );

/* Process */
sprintf(
    static system_string[],
    "echo \"alter table %s add %s %s;\" | "
    "tee_appaserver.sh | "
    "sql.e 2>&1",
    appaserver_table_name,
    attribute_name,
    attribute_database_datatype );

/* Usage */
char *add_column_process_filespecifiction(
    char *application_name,
    char *folder_name,
    char *attribute_name,
    char *data_directory );

/* Process */
snprintf(
    static process_filespecification[],
    sizeof ( process_filespecification ),
    "%s/%s/add_column_%s_%s.sh",
    data_directory,
    application_name,
    folder_name,
    attribute_name );

/* Driver */
if ( execute )
{
    char *results =
        string_pipe_fetch(
            add_column->
                execute_system_string );

    process_execution_count_increment(
        process_name );
}
else
{
    FILE *appaserver_output_file(
        add_column->process_filename );

    fprintf(    appaserver_output_file(),
                "%s\n",
                add_column->shell_script );

    fclose( appaserver_output_file() );

    shell_script_execute_bit_system_string(
        add_column->process_filespecification );

    system( shell_script_execute_bit_system_string() );
}

/* Attributes */
FOLDER_ATTRIBUTE *folder_attribute;
char *appaserver_table_name;
char *attribute_database_datatype;
char *execute_system_string;
char *shell_script;
char *process_filespecification;
```



ADD_COLUMN (2)

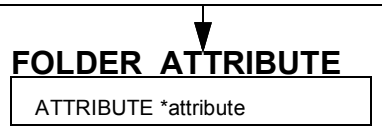
```
/* Usage */
char *add_column_shell_script(
    char *application_name,
    FOLDER_ATTRIBUTE *folder_attribute,
    char *appaserver_table_name(),
    char *add_column_execute_system_string() );

/* Process */
char shell_script[];
char *ptr = shell_script;

ptr += sprintf( ptr,
        "%s\n",
        shell_script_open_string(
            application_name ) );

ptr += sprintf( ptr,
        "(\n" );

ptr += sprintf( ptr,
        "%s\n",
        add_column_execute_system_string );

char *attribute_insert_statement(
    ATTRIBUTE_SELECT,
    ATTRIBUTE_TABLE,
    folder_attribute->attribute );

char *add_column_shell_script_system_string(
    attribute_insert_statement() );

ptr += sprintf( ptr,
        "%s\n",
        add_column_shell_script_system_string() );
```

```
char *folder_attribute_insert_statement(
    FOLDER_ATTRIBUTE_SELECT,
    FOLDER_ATTRIBUTE_TABLE,
    folder_attribute );

char *add_column_shell_script_system_string(
    folder_attribute_insert_statement() );

ptr += sprintf( ptr,
        "%s\n",
        add_column_shell_script_system_string() );

char *add_column_relation_insert_system_string(
    RELATION_TABLE,
    RELATION_PRIMARY_KEY,
    folder_attribute->folder_name,
    folder_attribute->attribute_name );

if ( add_column_relation_insert_system_string() )
{
    ptr += sprintf( ptr,
            "%s\n",
            add_column_relation_insert_system_string() );
}

ptr += sprintf( ptr,
        ") 2>&1 | grep -vi duplicate\n" );

ptr += sprintf( ptr,
        "%s\n",
        shell_script_close_string() );

/* Usage */
char *add_column_shell_script_system_string(
    char *statement );
```

```
/* Process */
sprintf(    char system_string[],
            "echo \"%s\" | sql.e 2>&1",
            statement );

/* Usage */
char *add_column_relation_insert_system_string(
    char *RELATION_TABLE,
    char *RELATION_PRIMARY_KEY,
    char *folder_attribute->folder_name,
    char *folder_attribute->attribute_name );

/* Process */
LIST *folder_attribute_fetch_primary_key_list(
    folder_name );

LIST *relation_mto1_list(
    folder_name,
    folder_attribute_fetch_primary_key_list() );

RELATION_MTO1 *relation_mto1_consumes(
    attribute_name,
    relation_mto1_list() );

char *relation_insert_statement(
    RELATION_TABLE,
    RELATION_PRIMARY_KEY,
    relation_mto1_consumes()->many_folder_name,
    relation_mto1_consumes()->one_folder_name,
    relation_mto1_consumes()->
        relation->
        related_attribute_name );
```

ALTER_DATATYPE (1)

```
/* Usage */
ALTER_DATATYPE *alter_datatype_new(
    char *application_name,
    char *sql_injection_escape(
        folder_name ),
    char *sql_injection_escape(
        attribute_name ),
    char *appaserver_parameter->data_directory );

/* Process */
ALTER_DATATYPE *alter_datatype_calloc(
    void );

FOLDER_ATTRIBUTE *folder_attribute =
    folder_attribute_fetch(
        folder_name,
        attribute_name,
        1 /* fetch_attribute */ );

char *appaserver_table_name(
    folder_name );

char *attribute_database_datatype(
    folder_attribute->attribute->datatype_name,
    folder_attribute->attribute->width,
    folder_attribute->attribute->float_decimal_places,
    folder_attribute->primary_key_index,
    folder_attribute->default_value );

char *alter_datatype_execute_system_string(
    attribute_name,
    appaserver_table_name(),
    attribute_database_datatype() );

char *alter_datatype_shell_script(
    application_name,
    folder_attribute,
    alter_datatype_execute_system_string() );

char *alter_datatype_process_filespecification(
    application_name,
    folder_name,
    attribute_name,
    data_directory );

/* Usage */
char *alter_datatype_execute_system_string(
    char *attribute_name,
    char *appaserver_table_name(),
    char *attribute_database_datatype() );

/* Process */
snprintf(
    static system_string[],
    sizeof ( system_string ),
    "echo \"alter table %s modify %s %s;\" |"
    "tee_appaserver.sh |"
    "sql.e 2>&1",
    appaserver_table_name,
    attribute_name,
    attribute_database_datatype );

/* Usage */
char *alter_datatype_process_filespecification(
    char *application_name,
    char *folder_name,
    char *attribute_name,
    char *data_directory );

/* Process */
snprintf(
    static process_filespecification[],
    sizeof ( process_filespecification ),
    "%s/%s/alter_datatype_%s_%s.sh",
    data_directory,
    application_name,
    folder_name,
    attribute_name );

/* Attributes */
FOLDER_ATTRIBUTE *folder_attribute;
char *appaserver_table_name;
char *attribute_database_datatype;
char *execute_system_string;
char *shell_script;
char *process_filespecification;
```



ALTER_DATATYPE (2)

```
/* Usage */
char *alter_datatype_shell_script(
    char *application_name,
    FOLDER_ATTRIBUTE *folder_attribute,
    char *alter_datatype_execute_system_string() );

/* Process */
char shell_script[];
char *ptr = shell_script;

ptr += sprintf( ptr,
        "%s\n",
        shell_script_open_string(
            application_name ) );

ptr += sprintf( ptr,
        "%s\n",
        alter_datatype_execute_system_string );

char *attribute_delete_statement(
    ATTRIBUTE_TABLE,
    folder_attribute->attribute_name );

char *alter_datatype_shell_script_system_string(
    attribute_delete_statement() );

ptr += sprintf( ptr,
        "%s\n",
        alter_datatype_shell_script_system_string() );

char *attribute_insert_statement(
    ATTRIBUTE_SELECT,
    ATTRIBUTE_TABLE,
    folder_attribute->attribute );

char *alter_datatype_shell_script_system_string(
    attribute_insert_statement() );

ptr += sprintf( ptr,
        "%s\n",
        alter_datatype_shell_script_system_string() );

ptr += sprintf( ptr,
        "%s\n",
        shell_script_close_string() );

return strdup( shell_script );

/* Usage */
char *alter_datatype_shell_script_system_string(
    char *statement );

/* Process */
snprintf(
    static char system_string[],
    sizeof ( system_string ),
    "echo \"%s\" | "
    "tee_appaserver.sh | "
    "sql.e 2>&1",
    statement );

/* Driver */
if ( execute )
{
    char *results =
        string_pipe_fetch(
            alter_datatype->
                execute_system_string );

    process_execution_count_increment(
        process_name );
}
else
{
    FILE *appaserver_output_file(
        alter_datatype->process_filespecification );

    fprintf(    appaserver_output_file(),
                "%s\n",
                alter_datatype->shell_script );

    fclose( appaserver_output_file() );

    char *shell_script_execute_bit_system_string(
        alter_datatype->process_filespecification );

    system( shell_script_execute_bit_system_string() );
}
```

RENAME_TABLE (1)

```
/* Usage */
RENAME_TABLE *rename_table_new(
    char *application_name,
    char *security_sql_injection_escape(
        old_folder_name ),
    char *security_sql_injection_escape(
        new_folder_name ),
    char *appaserver_parameter->data_directory );

/* Process */
RENAME_TABLE *rename_table_calloc(
    void );

char *old_appaserver_table_name =
    strdup(
        appaserver_table_name(
            old_folder_name ) );

char *new_appaserver_table_name =
    strdup(
        appaserver_table_name(
            new_folder_name ) );

char *rename_table_execute_system_string(
    old_appaserver_table_name,
    new_appaserver_table_name );

char *rename_table_drop_index_name(
    SQL_DELIMITER,
    old_appaserver_table_name );

char *rename_table_drop_index_system_string(
    char *new_appaserver_table_name,
    char *rename_table_drop_index_name() );

LIST *folder_attribute_fetch_primary_key_list(
    new_folder_name );

char *create_table_primary_index_system_string(
    CREATE_TABLE_UNIQUE_SUFFIX,
    folder_attribute_fetch_primary_key_list(),
    new_appaserver_table_name );

LIST *rename_table_update_statement_list(
    FOLDER_TABLE,
    ROLE_FOLDER_TABLE,
    RELATION_TABLE,
    FOLDER_ATTRIBUTE_TABLE,
    FOREIGN_ATTRIBUTE_TABLE,
    FOLDER_ROW_LEVEL_RESTRICTION_TABLE,
    PROCESS_PARAMETER_TABLE,
    FOLDER_OPERATION_TABLE,
    old_folder_name,
    new_folder_name );

char *rename_table_shell_script(
    application_name,
    rename_table_execute_system_string(),
    LIST *rename_table_update_statement_list() );

char *rename_table_process_filespecification(
    application_name,
    old_folder_name,
    data_directory );

/* Attributes */
char *old_appaserver_table_name;
char *new_appaserver_table_name;
char *execute_system_string;
char *drop_index_name;
char *drop_index_system_string;
char *create_table_primary_index_system_string;
LIST *update_statement_list;
char *shell_script;
char *process_filespecification;
```

RENAME_TABLE (2)

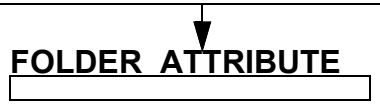
```
/* Usage */
char *rename_table_execute_system_string(
    char *old_appaserver_table_name,
    char *new_appaserver_table_name );

/* Process */
sprintf(
    static char system_string[],
    "echo \"alter table %s rename as %s\;" | "
    "tee_appaserver.sh | "
    "sql.e 2>&1",
    old_appaserver_table_name,
    new_appaserver_table_name );

/* Usage */
char *rename_table_process_filespecification(
    char *application_name,
    char *old_folder_name,
    char *data_directory );

/* Process */
char *shell_script_filespecification(
    "rename_table" /* PROCESS_LABEL */,
    application_name,
    old_folder_name,
    data_directory );

/* Usage */
char *rename_table_shell_script(
    char *application_name,
    char *rename_table_execute_system_string(),
    char *rename_table_drop_index_system_string(),
    char *create_table_primary_index_system_string(),
    LIST *rename_table_update_statement_list() );

/* Process */
char shell_script[];
char *ptr = shell_script;

ptr += sprintf( ptr,
        "%s\n",
        shell_script_open_string(
            application_name ) );

ptr += sprintf( ptr,
        "%s\n",
        rename_table_execute_system_string );

ptr += sprintf( ptr,
        "%s\n",
        rename_table_drop_index_system_string );

ptr += sprintf( ptr,
        "%s\n",
        create_table_primary_index_system_string );

for char *update_statement in
    rename_table_update_statement_list
{
    char *rename_table_system_string(
        update_statement );

    ptr += sprintf( ptr,
            "%s\n",
            rename_table_system_string() );
}

ptr += sprintf( ptr,
        "%s\n",
        shell_script_close_string() );

return strdup( shell_script );

/* Usage */
LIST *rename_table_update_statement_list(
    const char *FOLDER_TABLE,
    const char *ROLE_FOLDER_TABLE,
    const char *RELATION_TABLE,
    const char *FOLDER_ATTRIBUTE_TABLE,
    const char *FOREIGN_ATTRIBUTE_TABLE,
    const char *
        FOLDER_ROW_LEVEL_RESTRICTION_TABLE,
    const char *PROCESS_PARAMETER_TABLE,
    const char *FOLDER_OPERATION_TABLE,
    char *old_folder_name,
    char *new_folder_name );

/* Process */
LIST *statement_list = list_new();

char *rename_table_update_statement(
    const char *table,
    char *old_folder_name,
    char *new_folder_name,
    const char *folder_attribute_name );

/* Usage */
char *rename_table_system_string(
    char *statement );

/* Process */
snprintf(
    static char system_string[],
    sizeof ( system_string ),
    "echo \"%s\" | "
    "tee_appaserver.sh | "
    "sql.e 2>&1",
    statement );

/* Usage */
char *rename_table_drop_index_name(
    const char SQL_DELIMITER,
    char *old_appaserver_table_name );

/* Process */
static char *rename_table_show_index_system_string(
    SQL_DELIMITER,
    old_appaserver_table_name );

char *string_pipe_fetch(
    rename_table_show_index_system_string() );

/* Usage */
static char *rename_table_show_index_system_string(
    const char SQL_DELIMITER,
    char *old_appaserver_table_name );

/* Process */
snprintf(
    static system_string[],
    sizeof ( system_string ),
    "echo \"show index from %s;\" | "
    "sql.e '%c' |"
    "head -1 | "
    "piece.e %c 2",
    old_appaserver_table_name,
    SQL_DELIMITER );
```

RENAME_TABLE (3)

```
/* Driver */
if ( execute )
{
    char *result =
        string_pipe_fetch(
            rename_table->
                execute_system_string );

    if ( result ) printf( "<p>%s\n", result );

    system(
        rename_table->
            drop_index_system_string );

    system(
        rename_table->
            create_table_primary_index_system_string );

    process_increment_count(
        process_name );
}
else
{
    FILE *appaserver_output_file(
        rename_table->process_filespecification );

    fprintf(
        appaserver_output_file(),
        "%s\n",
        rename_table->shell_script );

    fclose( appaserver_output_file() );

    shell_script_execute_bit_system_string(
        rename_table->process_filespecifiction );

    system( shell_script_execute_bit_system_string() );
}
```

RENAME_COLUMN (1)

```
/* Usage */
RENAME_COLUMN *rename_column_new(
    char *application_name,
    char *security_sql_injection_escape(
        folder_name ),
    char *security_sql_injection_escape(
        old_attribute_name ),
    char *security_sql_injection_escape(
        new_attribute_name ),
    char *appaserver_parameter->data_directory );

/* Process */
RENAME_COLUMN *rename_column_calloc(\
    void );

FOLDER_ATTRIBUTE *folder_attribute =
    folder_attribute_fetch(
        folder_name,
        new_attribute_name,
        1 /* fetch_attribute */ );

char *appaserver_table_name(
    folder_name );

char *attribute_database_datatype(
    folder_attribute->attribute->datatype_name,
    folder_attribute->attribute->width,
    folder_attribute->attribute->float_decimal_places,
    folder_attribute->primary_key_index,
    folder_attribute->default_value );

char *rename_column_execute_system_string(
    old_attribute_name,
    new_attribute_name,
    appaserver_table_name(),
    attribute_database_datatype() );

LIST *rename_column_update_statement_list(
    ATTRIBUTE_TABLE,
    FOLDER_ATTRIBUTE_TABLE,
    RELATION_TABLE,
    FOREIGN_ATTRIBUTE_TABLE,
    ROW_SECURITY_ROLE_UPDATE_TABLE,
    ROLE_ATTRIBUTE_EXCLUDE_TABLE,
        old_attribute_name,
    new_attribute_name );

char *rename_column_shell_script(
    application_name,
    rename_column_execute_system_string(),
    LIST *rename_column_update_statement_list() );

char *rename_column_process_filespecification(
    application_name,
    folder_name,
    new_attribute_name,
    data_directory );

/* Attributes */
FOLDER_ATTRIBUTE *folder_attribute;
char *appaserver_table_name;
char *attribute_database_datatype;
char *execute_system_string;
LIST *update_statement_list;
char *shell_script;
char *process_filespecification;
```



RENAME_COLUMN (2)

```
/* Usage */
char *rename_column_execute_system_string(
    char *old_attribute_name,
    char *new_attribute_name,
    char *appaserver_table_name(),
    char *attribute_database_datatype() );

/* Process */
snprintf(
    static char system_string[],
    sizeof ( system_string ),
    "echo \"alter table %s change %s %s %s;\" | "
    "tee_appaserver.sh | "
    "sql.e 2>&1",
    appaserver_table_name,
    old_attribute_name,
    new_attribute_name,
    attribute_database_datatype );

/* Usage */
char *rename_column_process_filespecification(
    char *application_name,
    char *folder_name,
    char *new_attribute_name,
    char *data_directory );

/* Process */
snprintf(
    static char process_filespecification[],
    sizeof ( process_filespecification ),
    "%s/%s/rename_column_%s_%s.sh",
    data_directory,
    application_name,
    folder_name,
    new_attribute_name );

/* Usage */
char *rename_column_shell_script(
    char *application_name,
    char *rename_column_execute_system_string(),
    LIST *rename_column_update_statement_list() );

/* Process */
char shell_script[];
char *ptr = shell_script;

ptr += sprintf( ptr,
        "%s\n",
        shell_script_open_string(
            application_name ) );

ptr += sprintf( ptr,
        "%s\n",
        rename_column_execute_system_string );

for char *update_statement in
    rename_column_update_statement_list
{
    char *rename_column_system_string(
        update_statement );

    ptr += sprintf( ptr,
            "%s\n",
            rename_column_system_string() );
}

ptr += sprintf( ptr,
        "%s\n",
        shell_script_close_string() );

return strdup( shell_script );

/* Usage */
LIST *rename_column_update_statement_list(
    const char *ATTRIBUTE_TABLE,
    const char *FOLDER_ATTRIBUTE_TABLE,
    const char *RELATION_TABLE,
    const char *FOREIGN_ATTRIBUTE_TABLE,
    const char *ROW_SECURITY_ROLE_UPDATE_TABLE,
    const char *ROLE_ATTRIBUTE_EXCLUDE_TABLE,
    const char *PROCESS_PARAMETER_TABLE,
    char *old_attribute_name,
    char *new_attribute_name );

/* Process */
LIST *statement_list = list_new();

char *rename_column_update_statement(
    const char *table,
    char *old_attribute_name,
    char *new_attribute_name,
    const char *folder_attribute_name );

/* Usage */
char *rename_column_system_string(
    char *statement );

/* Process */
snprintf(
    static char system_string[],
    sizeof ( system_string ),
    "echo \"%s\" | "
    "tee_appaserver.sh | "
    "sql.e 2>&1",
    statement );

/* Driver */
if ( execute )
{
    char *results =
        string_pipe_fetch(
            rename_column->
                execute_system_string );

    process_increment_count(
        process_name );
}
else
{
    FILE *appaserver_output_file(
        rename_column->process_filespecification );

    fprintf(    appaserver_output_file(),
                "%s\n",
                rename_column->shell_script );

    fclose( appaserver_output_file() );

    shell_script_execute_bit_system_string(
        rename_column->process_filespecification );

    system( shell_script_execute_bit_system_string() );
}
```

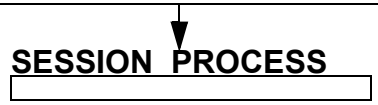
DROP_COLUMN (1)

```
/* Usage */
DROP_COLUMN *drop_column_new(
    int argc,
    char **argv,
    char *security_sql_injection_escape(
        application_name ),
    char *session_key,
    char *login_name,
    char *role_name,
    char *sql_injection_escape(
        folder_name ),
    char *sql_injection_escape(
        attribute_name ) );

/* Process */
DROP_COLUMN *drop_column_calloc(
    void );

/* Sets appaserver environment and outputs argv. */
/* Each parameter is security inspected. */
/* Any error will exit( 1 ). */
SESSION_PROCESS *session_process =
     session_process_integrity_exit(
          argc,
          argv,
          application_name,
          session_key,
          login_name,
          role_name,
          DROP_COLUMN_EXECUTABLE
               /* process_name */ );

char *appaserver_table_name(
    folder_name );

char *drop_column_execute_system_string(
    attribute_name,
    appaserver_table_name() );

LIST *drop_column_delete_statement_list(
    FOLDER_ATTRIBUTE_TABLE,
    FOREIGN_ATTRIBUTE_TABLE,
    ROW_SECURITY_ROLE_UPDATE_TABLE,
    folder_name,
    attribute_name );

char *drop_column_shell_script(
    application_name,
    drop_column_execute_system_string(),
    LIST *drop_column_delete_statement_list() );

char *appaserver_parameter_data_directory();

char *drop_column_process_filespecification(
    application_name,
    folder_name,
    attribute_name,
    appaserver_parameter_data_directory() );

/* Usage */
char *drop_column_execute_system_string(
    char *attribute_name,
    char *appaserver_table_name() );

/* Process */
snprintf(
    static char system_string[],
    sizeof ( system_string ),
    "echo \"alter table %s drop %s;\" |"
    "tee_appaserver.sh |"
    "sql.e 2>&1",
    appaserver_table_name,
    attribute_name );

/* Usage */
char *drop_column_process_filespecification(
    char *application_name,
    char *folder_name,
    char *attribute_name,
    char *data_directory );

/* Process */
snprintf(
    static char process_filespecification[],
    sizeof ( process_filespecification ),
    "%s/%s/drop_column_%s_%s.sh",
    data_directory,
    application_name,
    folder_name,
    attribute_name );

/* Usage */
char *drop_column_href_string(
    const char *DROP_COLUMN_EXECUTABLE,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name,
    char *attribute_name );

/* Process */
char *appaserver_parameter_cgi_directory();

char *application_server_address();

char *application_ssl_support_boolean(
    application_name );

char *application_http_prompt(
    appaserver_parameter_cgi_directory(),
    application_server_address(),
    application_ssl_support_boolean() );

snprintf(
    char href_string[],
    sizeof ( href_string ),
    "%s/%s?%s+%s+%s+%s+%s+%s+%s+%s+%s",
    application_http_prompt(),
    DROP_COLUMN_EXECUTABLE,
    application_name,
    session_key,
    login_name,
    role_name,
    DROP_COLUMN_EXECUTABLE /* process_name */,
    folder_name,
    attribute_name,
    "y" /* content_type_yn */,
    "y" /* execute_yn */ );
```

DROP_COLUMN (2)

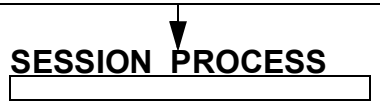
```
/* Usage */
char *drop_column_shell_script(
    char *application_name,
    char *drop_column_execute_system_string(),
    LIST *drop_column_delete_statement_list() );

/* Process */
char shell_script[];
char *ptr = shell_script;

ptr += sprintf( ptr,
        "%s\n",
        shell_script_open_string(
            application_name ) );

ptr += sprintf( ptr,
        "%s\n",
        drop_column_execute_system_string );

for char *delete_statement in
    drop_column_delete_statement_list
{
    char *drop_column_delete_statement_system_string(
        delete_statement );

    ptr += sprintf( ptr,
    "%s\n",
    drop_column_delete_statement_system_string() );
}

ptr += sprintf( ptr,
        "%s\n",
        shell_script_close_string() );

return strdup( shell_script );
```

```
/* Usage */
LIST *drop_column_delete_statement_list(
    const char *FOLDER_ATTRIBUTE_TABLE,
    const char *FOREIGN_ATTRIBUTE_TABLE,
    const char *ROW_SECURITY_ROLE_UPDATE_TABLE,
    char *folder_name,
    char *attribute_name );

/* Process */
LIST *statement_list = list_new();

char *drop_column_delete_statement(
    const char *table,
    char *folder_name,
    char *attribute_name,
    const char *column_folder_name,
    const char *column_attribute_name );

/* Usage */
char *drop_column_delete_statement_system_string(
    char *delete_statement );

/* Process */
snprintf(
    static char system_string[],
    sizeof ( system_string ),
    "echo \"%s\" | sql.e 2>&1",
    delete_statement );

/* Driver */
if ( execute )
{
    char *results =
        string_pipe_fetch(
            drop_column->
                execute_system_string );

    if ( results )
    {
        printf( "<p>%s\n", results );
    }

    process_increment_count(
        process_name );
}
else
{
    FILE *appaserver_output_file(
            drop_column->process_filespecification );

    fprintf(
        appaserver_output_file(),
        "%s\n",
        drop_column->shell_script );

    fclose( appaserver_output_file() );

    char *shell_script_execute_bit_system_string(
        drop_column->process_filespecification );

    system( shell_script_execute_bit_system_string() );
}

/* Attributes */
SESSION_PROCESS *session_process;
char *appaserver_table_name;
char *execute_system_string;
LIST *delete_statement_list;
char *shell_script;
char *appaserver_parameter_data_directory;
char *process_filespecification;
```



DROP_TABLE (1)

```
/* Usage */
DROP_TABLE *drop_table_new(
    int argc,
    char **argv,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *sql_injection_escape(
        table_name ) );

/* Process */
DROP_TABLE *drop_table_calloc(
    void );

/* Sets appaserver environment and outputs argv. */
/* Each parameter is security inspected. */
/* Any error will exit( 1 ). */
SESSION_PROCESS *session_process =
      session_process_integrity_exit(
            argc,
            argv,
            application_name,
            session_key,
            login_name,
            role_name,
            DROP_TABLE_EXECUTABLE
                  /* process_name */ );

char *drop_table_execute_system_string(
    table_name );

char *drop_table_shell_script(
    application_name,
    drop_table_execute_system_string() );

char *appaserver_parameter_data_directory();

char *drop_table_process_filespecification(
    application_name,
    table_name,
    appaserver_parameter_data_directory() );

/* Attributes */
SESSION_PROCESS *session_process;
char *execute_system_string;
char *shell_script;
char *appaserver_parameter_data_directory;
char *process_filespecification;
```



DROP_TABLE (2)

```
/* Usage */
char *drop_table_execute_system_string(
    char *table_name );

/* Process */
snprintf(
    static char system_string[],
    sizeof ( system_string ),
    "echo \"drop table %s;\" |"
    "tee_appaserver.sh |"
    "sql.e 2>&1"
    table_name );

/* Usage */
char *drop_table_process_filespecification(
    char *application_name,
    char *table_name,
    char *data_directory );

/* Process */
char *shell_script_filespecification(
    DROP_TABLE_EXECUTABLE /* PROCESS_LABEL */,
    application_name,
    table_name /* folder_name */,
    data_directory );

/* Usage */
char *drop_table_href_string(
    const char *DROP_TABLE_EXECUTABLE,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *table_name );

/* Process */
char *appaserver_parameter_cgi_directory();

char *application_server_address();

char *application_ssl_support_boolean(
    application_name );

char *application_http_prompt(
    appaserver_parameter_cgi_directory(),
    application_server_address(),
    application_ssl_support_boolean() );

snprintf(
    char href_string[],
    sizeof ( href_string ),
    "%s/%s?%s+%s+%s+%s+%s+%s+%s+%s+%s",
    application_http_prompt(),
    DROP_TABLE_EXECUTABLE,
    application_name,
    session_key,
    login_name,
    role_name,
    DROP_TABLE_EXECUTABLE /* process_name */,
    table_name,
    "y" /* content_type_yn */,
    "y" /* execute_yn */ );

/* Usage */
char *drop_table_shell_script(
    char *application_name,
    char *drop_table_execute_system_string() );

/* Process */
char shell_script[];
char *ptr = shell_script;

ptr += sprintf( ptr,
        "%s\n",
        shell_script_open_string(
            application_name ) );

ptr += sprintf( ptr,
        "%s\n",
        drop_table_execute_system_string );

ptr += sprintf( ptr,
        "%s\n",
        shell_script_close_string() );

return strdup( shell_script );

/* Driver */
if ( execute )
{
    char *results =
        string_pipe_fetch(
            drop_table->
                execute_system_string );

    printf( "<p>%s\n", results );

    process_increment_count(
        process_name );

    printf( "<h3>Process complete.</h3>\n" );
}
else
{
    FILE *appaserver_output_file(
        drop_table->process_filespecification );

    fprintf(
        appaserver_output_file(),
        "%s\n",
        drop_table->shell_script );

    fclose( appaserver_output_file() );

    char *shell_script_execute_bit_system_string(
        drop_table->process_filespecification );

    system( shell_script_execute_bit_system_string() );
}
```

EXPORT_SUBSCHEMA (1)

```
/* Usage */
EXPORT_SUBSCHEMA *export_subschema_new(
    char *application_name,
    char *folder_name_list_string,
    boolean exclude_roles_boolean,
    char *appaserver_parameter->data_directory );

/* Process */
EXPORT_SUBSCHEMA *export_subschema_calloc(
    void );

LIST *export_subschema_folder_name_list(
    SQL_DELIMITER,
    folder_name_list_string );

if ( !list_rewind( export_subschema_folder_name_list() )
{
    return NULL;
}

char *export_subschema_filespecification(
    char *application_name,
    char *data_directory,
    char *date_now_yyyy_mm_dd(
        date_utc_offset() ) );

FILE *export_subschema_process_file(
    char *export_subschema_filespecification() );

char *shell_script_open_string(
    application_name );

fprintf(    export_subschema_process_file(),
            "%s\n",
            shell_script_open_string() );

char *shell_script_here_open_string(
    SHELL_SCRIPT_HERE_LABEL );

fprintf(    export_subschema_process_file(),
            "%s\n",
            shell_script_here_open_string() );

for char *folder_name in
    export_subschema_folder_name_list()
{
    void export_subschema_output_folder(
        export_subschema_process_file(),
        folder_name );

    void export_subschema_output_relation(
        export_subschema_process_file(),
        folder_name );

    void export_subschema_output_standard(
        "table_column" /* appaserver_folder_name */,
        export_subschema_process_file(),
        folder_name );

    void export_subschema_output_attribute(
        export_subschema_process_file(),
        folder_name );

    void export_subschema_output_operation(
        export_subschema_process_file(),
        folder_name );

    void export_subschema_output_standard(
        "table_operation" /* appaserver_folder_name */,
        export_subschema_process_file(),
        folder_name );

    if ( !exclude_roles_boolean )
    {
        void export_subschema_output_standard(
            "role_table" /* appaserver_folder_name */,
            export_subschema_process_file(),
            folder_name );
    }

    void export_subschema_output_standard(
        "table_row_level_restriction"
            /* appaserver_folder_name */,
        export_subschema_process_file(),
        folder_name );

    void export_subschema_output_standard(
        "row_security_role_update"
            /* appaserver_folder_name */,
        export_subschema_process_file(),
        folder_name );

    void export_subschema_output_standard(
        "foreign_column" /* appaserver_folder_name */,
        export_subschema_process_file(),
        folder_name );

    void export_subschema_output_subschema(
        export_subschema_process_file(),
        folder_name );

    void export_subschema_output_process(
        export_subschema_process_file(),
        "post_change_process"
            /* process_column_name */,
        folder_name );

    void export_subschema_output_process(
        export_subschema_process_file(),
        "populate_drop_down_process"
            /* process_column_name */,
        folder_name );
}

char *shell_script_here_close_string(
    SHELL_SCRIPT_HERE_LABEL,
    0 /* not tee_appaserver_boolean */,
    0 /* not html_paragraph_wrapper_boolean */ );

fprintf(    export_subschema_process_file(),
            "%s\n",
            shell_script_here_close_string() );

char *shell_script_close_string();

fprintf(    export_subschema_process_file(),
            "%s\n",
            shell_script_close_string() );

fclose( export_subschema_process_file() );

char *shell_script_execute_bit_system_string(
    export_subschema_filespecification()
        /* shell_script_filespecification */ );

/* Attributes */
LIST *folder_name_list;
char *filespecification;
char *shell_script_open_string;
char *shell_script_here_open_string;
char *shell_script_here_close_string;
char *shell_script_close_string;
char *shell_script_execute_bit_system_string;
```

EXPORT_SUBSCHEMA (2)

```
/* Usage */
LIST *export_subschema_folder_name_list(
    const char SQL_DELIMITER,
    char *folder_name_list_string );

/* Process */
LIST *list_string_list(
    folder_name_list_string,
    SQL_DELIMITER );

/* Usage */
char *export_subschema_output_folder(
    FILE *export_subschema_process_file(),
    char *folder_name );

/* Process */
char *export_subschema_select(
    “table” /* folder_name */ );

char *folder_primary_where_string(
    FOLDER_PRIMARY_KEY,
    folder_name );

char *appaserver_system_string(
    export_subschema_select(),
    FOLDER_TABLE,
    folder_primary_where_string() );

char *string_pipe_input(
    appaserver_system_string() );

char *insert_folder_string_sql_statement(
    SQL_DELIMITER,
    FOLDER_TABLE,
    export_subschema_select()
        /* attribute_name_list_string */,
    string_pipe_input() /* delimited_string */ );

fprintf(    export_subschema_process_file,
            “%s\n”,
            insert_folder_string_sql_statement() );

/* Usage */
void export_subschema_output_relation(
    FILE *export_subschema_process_file(),
    char *folder_name );

/* Process */
char *export_subschema_select(
    “relation” /* folder_name */ );

char *relation_mto1_where_string(
    FOLDER_PRIMARY_KEY,
    folder_name /* many_folder_name */ );

char *appaserver_system_string(
    export_subschema_select(),
    RELATION_TABLE,
    relation_mto1_where_string() );

FILE *export_subschema_input_pipe(
    char *appaserver_system_string() )

for char input[] in
    string_input(
        export_subschema_input_pipe() ) )
{
    char *insert_folder_string_sql_statement(
        SQL_DELIMITER,
        RELATION_TABLE,
        export_subschema_select()
            /* attribute_name_list_string */,
        input /* delimited_string */ );

    fprintf(    export_subschema_process_file,
                “%s\n”,
                insert_folder_string_sql_statement() );
}

pclose( export_subschema_input_pipe() );

char *relation_one2m_where_string(
    RELATION_FOREIGN_KEY
        /* probably related_table */,
    folder_name /* one_folder_name */ );

char *appaserver_system_string(
    export_subschema_select(),
    RELATION_TABLE,
    relation_one2m_where_string() );

FILE *export_subschema_input_pipe(
    appaserver_system_string() );

for char input[] in
    string_input(
        export_subschema_input_pipe() ) )
{
    char *insert_folder_string_sql_statement(
        SQL_DELIMITER,
        RELATION_TABLE,
        export_subschema_select()
            /* attribute_name_list_string */,
        input /* delimited_string */ );

    fprintf(    export_subschema_process_file,
                “%s\n”,
                insert_folder_string_sql_statement() );
}

pclose( export_subschema_input_pipe() );
```

EXPORT_SUBSCHEMA (3)

```
/* Usage */
void export_subschema_output_attribute(
    FILE *export_subschema_process_file(),
    char *folder_name );

/* Process */
char *export_subschema_select(
    “column” /* folder_name */ );

char *export_subschema_attribute_where(
    ATTRIBUTE_PRIMARY_KEY,
    folder_name );

char *appaserver_system_string(
    export_subschema_select(),
    ATTRIBUTE_TABLE,
    export_subschema_attribute_where() );

FILE *export_subschema_input_pipe(
    appaserver_system_string() );

for char input[] in
    string_input(
        export_subschema_input_pipe() ) )
{
    char *insert_folder_string_sql_statement(
        SQL_DELIMITER,
        ATTRIBUTE_TABLE,
        export_subschema_select()
            /* attribute_name_list_string */,
        input /* delimited_string */ );

    fprintf(    export_subschema_process_file,
                “%s\n”,
                insert_folder_string_sql_statement() );
}

pclose( export_subschema_input_pipe() );

/* Usage */
void export_subschema_output_subschema(
    FILE *export_subschema_process_file(),
    char *folder_name );

/* Process */
static LIST *subschema_list = {0};

if ( !subschema_list ) subschema_list = list_new();

char *folder_primary_where_string(
    FOLDER_PRIMARY_KEY,
    folder_name );

char *appaserver_system_string(
    “subschema” /* select */,
    FOLDER_TABLE,
    folder_primary_where_string() );

char *subschema =
    char *string_pipe_input(
        appaserver_system_string() );

boolean list_string_boolean(
    subschema,
    subschema_list );

if ( list_string_boolean() ) return;

list_set( subschema_list, subschema );

char *insert_folder_string_sql_statement(
    SQL_DELIMITER,
    “subschema” /* table_name */,
    “subschema” /* attribute_name_list_string */,
    subschema /* delimited_string */ );

fprintf(    export_subschema_process_file,
            “%s\n”,
            insert_folder_string_sql_statement() );
```

EXPORT_SUBSCHEMA (4)

```
/* Usage */
void export_subschema_output_operation(
    FILE *export_subschema_process_file(),
    char *folder_name );

/* Process */
void export_subschema_output_standard(
    “table_operation” /* appaserver_folder_name */,
    export_subschema_process_file(),
    folder_name );

LIST *folder_operation_list(
    folder_name,
    (char *)0 /* role_name */,
    1 /* fetch_operation */
    1 /* fetch_process */ );

for FOLDER_OPERATION *folder_operation in
    folder_operation_list()
{
    char *export_subschema_operation_string(
        const char SQL_DELIMITER,
        char *folder_operation->
            operation->
            operation_name,
        boolean folder_operation->
            operation->
            output_boolean );

char *insert_folder_string_sql_statement(
    SQL_DELIMITER,
    OPERATION_TABLE,
    “operation,output_yn”
        /* attribute_name_list_string */,
    export_subschema_operation_string()
        /* delimited_string */ );

fprintf(    export_subschema_process_file,
            “%s\n”,
            insert_folder_string_sql_statement() );

char *export_subschema_process_string(
    SQL_DELIMITER,
    folder_operation->
        operation->
        operation_name
            /* process_name */,
        folder_operation->
            operation->
            process->
            command_line );

static char *export_process_escape_command_line(
    export_subschema_process_string() );

char *insert_folder_string_sql_statement(
    SQL_DELIMITER,
    PROCESS_TABLE,
    “process,command_line”
        /* attribute_name_list_string */,
    export_process_escape_command_line()
        /* delimited_string */ );

fprintf(    export_subschema_process_file,
            “%s\n”,
            insert_folder_string_sql_statement() );
}
```

EXPORT_SUBSCHEMA (5)

```
/* Usage */
void export_subschema_output_process(
    FILE *export_subschema_process_file(),
    const char *process_column_name,
    char *folder_name );

/* Process */
char *folder_column_fetch(
    process_column_name,
    folder_name );

if ( folder_column_fetch() )
{
    char *process_command_line_fetch(
        folder_column_fetch() /* process_name */ );

    static char *export_process_escape_command_line(
        process_command_line_fetch() );

    char *export_subschema_process_string(
        const char SQL_DELIMITER,
        char *folder_column_fetch()
            /* process_name */,
        char *export_process_escape_command_line() );

    char *insert_folder_string_sql_statement(
        SQL_DELIMITER,
        PROCESS_TABLE,
        "process,command_line"
            /* attribute_name_list_string */,
        string_escape_quote_dollar()
            /* delimited_string */ );

    fprintf(    export_subschema_process_file,
                "%s\n",
                insert_folder_string_sql_statement() );
}

/* Usage */
void export_subschema_output_standard(
    const char *appaserver_folder_name,
    FILE *export_subschema_process_file(),
    char *folder_name );

/* Process */
char *export_subschema_select(
    appaserver_folder_name );

char *appaserver_table_name(
    appaserver_folder_name );

char *folder_primary_where_string(
    FOLDER_PRIMARY_KEY,
    folder_name );

char *appaserver_system_string(
    export_subschema_select(),
    appaserver_table_name(),
    folder_primary_where_string() );

FILE *export_subschema_input_pipe(
    appaserver_system_string() )

for char input[] in
    string_input(
        export_subschema_input_pipe() ) )
{
    char *insert_folder_string_sql_statement(
        SQL_DELIMITER,
        appaserver_table_name(),
        export_subschema_select()
            /* attribute_name_list_string */,
        input /* delimited_string */ );

    fprintf(    export_subschema_process_file,
                "%s\n",
                insert_folder_string_sql_statement() );
}

pclose( export_subschema_input_pipe() );

/* Usage */
char *export_subschema_select(
    char *appaserver_folder_name );

/* Process */
return
char *folder_attribute_list_string(
    appaserver_folder_name );

/* Usage */
char *export_subschema_attribute_where(
    const char *ATTRIBUTE_PRIMARY_KEY,
    char *folder_name );

/* Process */
LIST *folder_attribute_list(
    folder_name,
    (LIST *)0 /* exclude_attribute_name_list */,
    0 /* not fetch_attribute */,
    0 /* not cache_boolean */ );

LIST *folder_attribute_name_list(
    (char *)0 /* folder_name */,
    folder_attribute_list() );

char *list_single_quotes_around_string(
    folder_attribute_name_list() );

snprintf(   char where[],
            sizeof ( where ),
            "%s in (%s)",
            ATTRIBUTE_PRIMARY_KEY,
            list_single_quotes_around_string() );

free( list_single_quotes_around_string() );

/* Driver */
system( export_subschema->
    shell_script_execute_bit_system_string );

process_increment_count(
    process_name );

printf(
    "<h3>Created: %s</h3>\n",
    export_subschema->filename );
```

EXPORT_PROCESS (1)

```
/* Usage */
FILE *export_process_stream(
    char *filename,
    const char *mode );

/* Process */
FILE *fopen( filename, mode );

/* Usage */
void export_process_output_process(
    FILE *export_process_stream(),
    char *process_name );

/* Process */
char *process_primary_where(
    process_name );

char *export_process_delete_sql_statement(
    PROCESS_TABLE,
    process_primary_where()
        /* where_string */ );

fprintf(
    export_process_stream,
    "%s\n",
    export_process_delete_sql_statement() );

char *export_process_select(
    PROCESS_TABLE /* folder_name */ );

char *appaserver_system_string(
    export_process_select(),
    PROCESS_TABLE,
    process_primary_where() );

char *string_pipe_input(
    appaserver_system_string() );

char *export_process_escape(
    char *string_pipe_input() );

char *insert_folder_string_sql_statement(
    SQL_DELIMITER,
    PROCESS_TABLE,
    export_process_select()
        /* attribute_name_list_string */,
    export_process_escape()
        /* delimited_string */ );

fprintf(
    export_process_stream,
    "%s\n",
    insert_folder_string_sql_statement() );

/* Usage */
void export_process_output_standard(
    char *table_name,
    FILE *export_process_stream(),
    char *process_name );

/* Process */
char *process_primary_where(
    process_name );

char *export_process_delete_sql_statement(
    PROCESS_PARAMETER_TABLE,
    process_primary_where()
        /* where_string */ );

fprintf(
    export_process_stream,
    "%s\n",
    export_process_delete_sql_statement() );

char *export_process_select(
    table_name /* folder_name */ );

char *appaserver_system_string(
    export_process_select(),
    table_name,
    process_primary_where() );

FILE *export_process_input_pipe(
    char *appaserver_system_string() )

for char input[] in
    string_input(
        export_process_input_pipe() ) )
{
    char *insert_folder_string_sql_statement(
        SQL_DELIMITER,
        table_name,
        export_process_select()
            /* attribute_name_list_string */,
        input /* delimited_string */ );

    fprintf(
        export_process_stream,
        "%s\n",
        insert_folder_string_sql_statement() );
}

pclose( export_process_input_pipe() );
```

EXPORT_PROCESS (2)

```
/* Usage */
void export_process_output_prompt(
    FILE *export_process_stream(),
    LIST *process_parameter_list() );

/* Process */
char *export_process_select(
    PROCESS_PARAMETER_PROMPT_TABLE );

char *export_process_prompt_where(
    process_parameter_list );

char *appaserver_system_string(
    export_process_select(),
    PROCESS_PARAMETER_PROMPT_TABLE,
    export_process_prompt_where() );

FILE *export_process_input_pipe(
    appaserver_system_string() );

for char input[] in
    string_input(
        export_process_input_pipe() ) )
{
    char *insert_folder_string_sql_statement(
        SQL_DELIMITER,
        PROCESS_PARAMETER_PROMPT_TABLE,
        export_process_select()
            /* attribute_name_list_string */,
        input /* delimited_string */ );

    fprintf(
        export_process_stream,
        "%s\n",
        insert_folder_string_sql_statement() );
}

pclose( export_process_input_pipe() );

/* Usage */
void export_process_output_drop_down_prompt(
    FILE *export_process_stream(),
    LIST *process_parameter_list() );

/* Process */
char *export_process_select(
 PROCESS_PARAMETER_DROP_DOWN_PROMPT_TABLE
        /* table_name */ );

char *export_process_drop_down_prompt_where(
    process_parameter_list );

char *appaserver_system_string(
 export_process_select(),
 PROCESS_PARAMETER_DROP_DOWN_PROMPT_TABLE,
 export_process_prompt_where() );

FILE *export_process_input_pipe(
    appaserver_system_string() );

for char input[] in
    string_input(
        export_process_input_pipe() ) )
{
 char *insert_folder_string_sql_statement(
  SQL_DELIMITER,
 PROCESS_PARAMETER_DROP_DOWN_PROMPT_TABLE,
  export_process_select()
        /* attribute_name_list_string */,
  input /* delimited_string */ );

    fprintf(
        export_process_stream,
        "%s\n",
        insert_folder_string_sql_statement() );
}

pclose( export_process_input_pipe() );

/* Usage */
char *export_process_drop_down_prompt_where(
    LIST *process_parameter_list() );

/* Process */
LIST *process_parameter_drop_down_prompt_list(
    process_parameter_list );

char *list_single_quotes_around_string(
    process_parameter_drop_down_prompt_list() );

snprintf(
    char where[],
    sizeof ( where ),
    "drop_down_prompt in (%s)",
    list_single_quotes_around_string() );

free( list_single_quotes_around_string() );
```

EXPORT_PROCESS (3)

```
/* Usage */
void export_process_output_drop_down_prompt_data(
    FILE *export_process_stream(),
    LIST *process_parameter_list );

/* Process */
char *export_process_select(
    PROCESS_PARAMETER_DROP_DOWN_PROMPT_DATA_TABLE );

char *export_process_drop_down_prompt_data_where(
    process_parameter_list );

char *appaserver_system_string(
    export_process_select(),
    PROCESS_PARAMETER_DROP_DOWN_PROMPT_DATA_TABLE,
    export_process_prompt_data_where() );

FILE *export_process_input_pipe(
    appaserver_system_string() );

for char input[] in
    string_input(
        export_process_input_pipe() ) )
{
    char *insert_folder_string_sql_statement(
        SQL_DELIMITER,
        PROCESS_PARAMETER_DROP_DOWN_PROMPT_DATA_TABLE,
        export_process_select()
            /* attribute_name_list_string */,
        input /* delimited_string */ );

    fprintf(      export_process_stream,
                  "%s\n",
                  insert_folder_string_sql_statement() );
}

pclose( export_process_input_pipe() );

/* Usage */
char *export_process_drop_down_prompt_data_where(
    LIST *process_parameter_list );

/* Process */
LIST *process_parameter_drop_down_prompt_list(
    process_parameter_list() );

char *list_single_quotes_around_string(
    process_parameter_drop_down_prompt_list() );

snprintf(
    subquery[],
    sizeof ( subquery ),
    "select drop_down_prompt "
    "from drop_down_prompt "
    "where drop_down_prompt in (%s)",
    list_single_quotes_around() );

snprintf(
    where[],
    sizeof ( where ),
    "drop_down_prompt in (%s)",
    subquery );

free( list_single_quotes_around_string() );
```

EXPORT_PROCESS (4)

```
/* Usage */
char *export_process_prompt_where(
    LIST *process_parameter_list() );

/* Process */
LIST *process_parameter_prompt_list(
    process_parameter_list );

char *list_single_quotes_around_string(
    process_parameter_prompt_list() );

snprintf(
    char where[],
    sizeof ( where ),
    "prompt in (%s)",
    list_single_quotes_around_string() );

free( list_single_quotes_around_string() );

/* Usage */
void export_process_output_process_group(
    FILE *export_process_stream(),
    char *process_name );

/* Process */
char *process_primary_where(
    process_name );

char *appaserver_system_string(
    "process_group" /* select */,
    PROCESS_TABLE,
    process_primary_where() );

char *string_pipe_input(
    appaserver_system_string() );

if ( *string_pipe_input() )
{
    char *insert_folder_string_sql_statement(
        SQL_DELIMITER,
        PROCESS_PARAMETER_GROUP_TABLE,
        "process_group" /* attribute_name_list_string */,
        string_pipe_input() /* delimited_string */ );

    fprintf(      export_process_stream,
                  "%s\n",
                  insert_folder_string_sql_statement() );
}

/* Usage */
char *export_process_select(
    char *folder_name );

/* Process */
return
char *folder_attribute_list_string( folder_name );

/* Usage */
char *export_process_delete_sql_statement(
    char *table_name,
    char *where_string );

/* Process */
snprintf(
    static sql_statement[],
    sizeof ( sql_statement ),
    "delete from %s where %s;",
    table_name,
    where_string );

/* Usage */
void export_process_output_populate_name(
    FILE *export_process_stream(),
    LIST *process_parameter_list(),
    boolean helper_name_boolean );

/* Process */
char *attribute_name_list_string =
    "process,command_line";

for PROCESS_PARAMETER *process_parameter in
    process_parameter_list
{
    if ( helper_name_boolean )
    {
        char *populate_process =
            process_parameter->
                populate_helper_process_name;
    }
    else
    {
        char *populate_process =
            process_parameter->
                populate_drop_down_process_name;
    }

    if ( !populate_process ) continue;

    char *process_command_line_fetch(
        populate_process /* process_name */ );

    static char *export_process_escape_command_line(
        process_command_line_fetch() );

    snprintf(
        delimited_string[],
        sizeof ( delimited_string ),
        "%s%c%s",
        populate_process,
        SQL_DELIMITER,
        export_process_escape_command_line() );

    char *insert_folder_string_sql_statement(
        SQL_DELIMITER,
        PROCESS_TABLE,
        attribute_name_list_string,
        delimited_string );

    fprintf(
        export_process_stream,
        "%s\n",
        insert_folder_string_sql_statement() );

    free( insert_folder_string_sql_statement() );
}

/* Usage */
static char *export_process_escape_command_line(
    char *process_command_line_fetch );

/* Process */
char *string_escape_quote_dollar(
    static destination[],
    process_command_line_fetch );
```

EXPORT_PROCESS_SHELL (1)

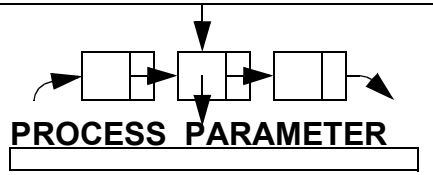
```
/* Usage */
EXPORT_PROCESS_SHELL *export_process_shell_new(
    char *application_name,
    char *export_process_name,
    boolean exclude_roles_boolean,
    char *appaserver_parameter->data_directory );

/* Process */
EXPORT_PROCESS_SHELL *
    export_process_shell_calloc(
        void );

LIST *export_process_shell_name_list(
    char *export_process_name,
    char delimiter /* ',' */ );

if ( !list_rewind(
        export_process_shell_name_list() ) )
{
        return NULL;
}

char *export_process_shell_filespecification(
    char *application_name,
    char *data_directory,
    char *date_now_yyyy_mm_dd( date_utc_offset() ) );

FILE *export_process_stream(
    export_process_shell_filespecification(),
    "w" /* mode */ );

char *shell_script_open_string( application_name );

fprintf(    export_process_stream(),
            "%s\n",
            shell_script_open_string() );

char *shell_script_here_open_string(
    SHELL_SCRIPT_HERE_LABEL );

fprintf(    export_process_stream(),
            "%s\n",
            shell_script_here_open_string() );

for char *process_name in
    export_process_shell_name_list()
{
    export_process_output_process(
        export_process_stream(),
        process_name );

    if ( !export_process_exclude_roles_boolean() )
    {
        export_process_output_standard(
            ROLE_PROCESS_TABLE,
            export_process_stream(),
            process_name );
    }

    export_process_output_standard(
        PROCESS_PARAMETER_TABLE,
        export_process_stream(),
        process_name );

    LIST *process_parameter_list(
        process_name,
        (char *)0 /* process_set_name */,
        0 /* not is_drillthru */ );

    export_process_output_prompt(
        export_process_stream(),
        process_parameter_list() );

    export_process_output_drop_down_prompt(
        export_process_stream(),
        process_parameter_list() );

    export_process_output_drop_down_prompt_data(
        export_process_stream(),
        process_parameter_list() );

    export_process_output_process_group(
        export_process_stream(),
        process_name );

    export_process_output_populate_name(
        export_process_stream(),
        process_parameter_list(),
        0 /* not helper_name_boolean */ );

    export_process_output_populate_name(
        export_process_stream(),
        process_parameter_list(),
        1 /* helper_name_boolean */ );
}

char *shell_script_here_close_string(
    SHELL_SCRIPT_HERE_LABEL,
    0 /* not tee_appaserver_boolean */,
    0 /* not html_paragraph_wrapper_boolean */ );

fprintf(    export_process_stream(),
            "%s\n",
            shell_script_here_close_string() );

char *shell_script_close_string();

fprintf(    export_process_stream(),
            "%s\n",
            shell_script_close_string() );

fclose( export_process_stream() );

char *shell_script_execute_bit_system_string(
    char *export_process_shell_filespecification() );
```

EXPORT_PROCESS_SHELL (2)

```
/* Driver */
system(
    export_process->
        shell_script_execute_bit_system_string );

process_increment_count( process_name );
```

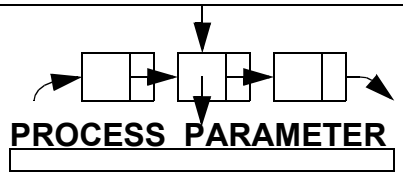
```
printf(
    "<h3>Created: %s</h3>\n",
    export_process->filename );

/* Attributes */
LIST *name_list;
char *filespecification;
```

```
LIST *process_parameter_list;
char *shell_script_open_string;
char *shell_script_here_open_string;
char *shell_script_here_close_string;
char *shell_script_close_string;
char *shell_script_execute_bit_system_string;
```



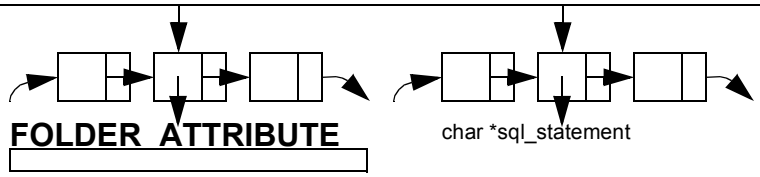
EXPORT_PROCESS_FILE

```
/* Usage */
EXPORT_PROCESS_FILE *export_process_file_new(
    char *filename,
    LIST *process_name_list );

/* Process */
EXPORT_PROCESS_FILE *
    export_process_filel_calloc(
        void );

FILE *export_process_stream(
    filename,
    “a” /* mode */ );

for char *process_name in process_name_list
{
    export_process_output_process(
        export_process_stream(),
        process_name );

    export_process_output_standard(
        PROCESS_PARAMETER_TABLE,
        export_process_stream(),
        process_name );

    LIST *process_parameter_list(
        process_name,
        0 /* not is_drillthru */ );

    export_process_output_prompt(
        export_process_stream(),
        process_parameter_list() );

    export_process_output_drop_down_prompt(
        export_process_stream(),
        process_parameter_list() );

    export_process_output_drop_down_prompt_data(
        export_process_stream(),
        process_parameter_list() );

    export_process_output_process_group(
        export_process_stream(),
        process_name );
}

fclose( export_process_stream() );

/* Attribute */
LIST *process_parameter_list;
```



CLONE_FOLDER (1)

```
/* Usage */
CLONE_FOLDER *clone_folder_new(
    char *application_name,
    char *sql_injection_escape(
        folder_name ),
    char *sql_injection_escape(
        attribute_name ),
    char *sql_injection_escape(
        old_data ),
    char *sql_injection_escape(
        new_data ),
    boolean delete_boolean,
    char *appaserver_parameter->data_directory );

/* Process */
CLONE_FOLDER *clone_folder_calloc(
    void );

LIST *sql_statement_list = list_new();

char *appaserver_table_name(
appaserver_table_namefolder_name );

if ( delete_boolean )
{
    char *clone_folder_where(
        attribute_name,
        new_data );

    char *clone_folder_delete_statement(
        char *appaserver_table_name(),
        char *clone_folder_where() );

    list_set(
        sql_statement_list,
        clone_folder_delete_statement() );

    return this;
}

LIST *folder_attribute_list(
    folder_name,
    (LIST *)0 /* exclude_attribute_name_list */,
    0 /* not fetch_attribute */,
    0 /* not cache_boolean */ );

LIST *folder_attribute_name_list(
    (char *)0 /* folder_name */,
    folder_attribute_list() );

int clone_folder_piece_offset(
    attribute_name,
    folder_attribute_name_list() );

char *clone_folder_select(
    folder_attribute_name_list() );

char *clone_folder_where(
    attribute_name,
    old_data );

char *appaserver_system_string(
    clone_folder_select(),
    appaserver_table_name(),
    clone_folder_where() );

FILE *clone_folder_input_pipe(
    char *appaserver_system_string() );

for char input[] in string_input(
    clone_folder_input_pipe() ) )
{
    void piece_replace(
        input /* source_destination */,
        SQL_DELIMITER,
        new_data,
        clone_folder_piece_offset() );

    list_set(
        sql_statement_list,
        char *insert_folder_string_sql_statement(
            SQL_DELIMITER,
            appaserver_table_name(),
            clone_folder_select()
                /* attribute_name_list_string */,
            input /* delimited_string */ ) );
}

pclose( clone_folder_input_pipe() );

char *clone_folder_filename(
    char *folder_name,
    char *data_directory,
    char *date_now_yyyy_mm_dd(
        date_utc_offset() );

void shell_script_spool_string_list(
    application_name,
    clone_folder_filename(),
    sql_statement_list /* string_list */ );

/* Attributes */
LIST *sql_statement_list;
char *appaserver_table_name;
char *delete_statement;
LIST *folder_attribute_list;
LIST *folder_attribute_name_list;
int piece_offset;
char *select;
char *where;
char *appaserver_system_string;
FILE *input_pipe;
char *filename;
```

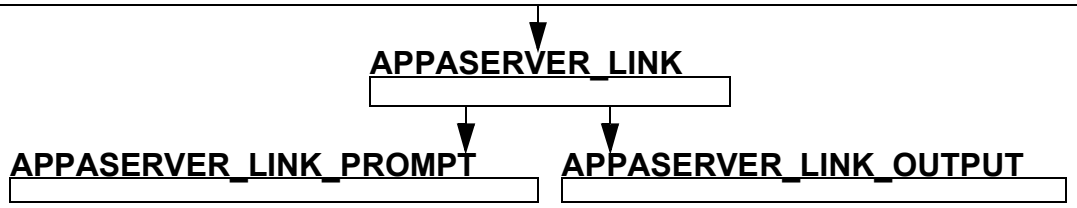


CLONE_FOLDER (2)

```
/* Usage */
int clone_folder_piece_offset(
    char *attribute_name,
    LIST *folder_attribute_name_list() );

/* Process */
int list_seek_offset(
    attribute_name /* item */,
    folder_attribute_name_list );

/* Usage */
char *clone_folder_select(
    LIST *folder_attribute_name_list() );

/* Process */
char *list_delimited( folder_attribute_name_list, ',' );

/* Usage */
char *clone_folder_where(
    char *attribute_name,
    char *data );

/* Process */
snprintf(
    static char where[],
    sizeof ( where ),
    "%s = '%s'",
    attribute_name,
    data );

/* Driver */
void clone_folder_display(
    LIST *clone_folder->sql_statement_list );

/* Process */
void html_string_list_stdout(
     HTML_TABLE_QUEUE_TOP_BOTTOM,
     "" /* title */,
     "SQL Statement" /* heading_string */,
     sql_statement_list
          /* string_list */ );

/* Driver */
void clone_folder_execute(
    LIST *clone_folder->sql_statement_list );

/* Process */
char *clone_folder_execute_system_string(
    void );

FILE *clone_folder_execute_pipe(
    char *clone_folder_execute_system_string );

for char *sql_statement in sql_statement_list
{
    fprintf(     clone_folder_execute_pipe(),
                 "%s\n",
                 sql_statement );
}

pclose( clone_folder_execute_pipe() );

process_increment_count( process_name );
```

EXPORT_TABLE (1)

```
/* Usage */
EXPORT_TABLE *export_table_new(
    char *application_name,
    char *folder_name,
    char *appaserver_parameter->data_directory,
    char *where );

/* Process */
EXPORT_TABLE *export_table_calloc(
    void );

char *application_table_name(
    folder_name );

LIST *folder_attribute_list(
    folder_name,
    (LIST *)0 /* exclude_attribute_name_list */,
    0 /* not fetch_attribute */,
    0 /* not cache_boolean */ );

LIST *folder_attribute_name_list(
    (char *)0 /* folder_name */,
    folder_attribute_list() );

if ( !list_length( folder_attribute_name_list() )
{
    return (EXPORT_TABLE *)0;
}

char *appaserver_select(
    folder_attribute_name_list() );

char *appaserver_system_string(
    appaserver_select(),
    application_table_name(),
    where );

if ( data_directory )
{
    char *date_now_yyyy_mm_dd(
        date_utc_offset() );

    char *export_table_shell_filespecification(
        char *application_name,
        char *folder_name,
        char *data_directory,
        char *date_now_yyyy_mm_dd() );

    char *export_table_execute_bit_system_string(
        char *export_table_shell_filespecification() );

     char *appaserver_spreadsheet_heading_string(
         folder_attribute_name_list() );
}

char *export_table_html_title(
    char *folder_name );

char *html_table_system_string(
     HTML_TABLE_QUEUE_TOP_BOTTOM,
     SQL_DELIMITER,
     export_table_html_title(),
     appaserver_select() /* heading_string */ );

char *insert_folder_statement_system_string(
    SQL_DELIMITER,
    application_table_name(),
    appaserver_select() /* field_list_string */ );

if (    application_name
&&  data_directory )
{
    APPASERVER_LINK *appaserver_link =
        appaserver_link_new(
            application_http_prefix(
                application_ssl_support_boolean(
                    application_name ) ),
            application_server_address(),
            data_directory,
            folder_name /* filename_stem */,
            application_name,
            getpid() /* process_id */,
            (char *)0 /* session_key */,
            date_now_yyyy_mm_dd() /* begin_date_string */,
            (char *)0 /* end_date_string */,
            "csv" /* extension */ );

     char *appaserver_link_anchor_html(
            appaserver_link->
                appaserver_link_prompt->
                filename,
            "_new" /* target_window */,
            "Link to csv file" /* prompt_message */ );

    char *export_table_spreadsheet_filespecification(
        char *appaserver_link->
            appaserver_link_output->
            filename );

    char *strdup(
        appaserver_spreadsheet_output_system_string(
            SQL_DELIMITER,
            export_table_spreadsheet_filename() ) );
}

/* Attributes */
char *application_table_name;
LIST *folder_attribute_name_list;
char *appaserver_select;
char *appaserver_system_string;
char *date_now_yyyy_mm_dd;
char *shell_filespecificationn;
char *execute_bit_system_string;
char *appaserver_spreadsheet_heading_string;
char *html_table_system_string;
char *insert_folder_statement_system_string;
APPASERVER_LINK *appaserver_link;
char *appaserver_link_anchor_html;
char *spreadsheet_filespecification;
char *appaserver_spreadsheet_output_system_string;
```



EXPORT_TABLE (2)

```
/* Usage */
LIST *export_table_spreadsheet_filename_list(
    LIST *export_table_list );

/* Process */
LIST *filename_list = list_new();

for EXPORT_TABLE *export_table in
    export_table_list )
{
    list_set(
        filename_list,
        export_table->spreadsheet_filespecification );
}

/* Usage */
LIST *export_table_shell_filename_list(
    LIST *export_table_list );

/* Process */
LIST *filename_list = list_new();

for EXPORT_TABLE *export_table in
    export_table_list )
{
    list_set(
        filename_list,
        export_table->shell_filespecification );
}
```

```
/* Driver */
void export_table_shell_output(
    char *application_name,
    char *export_table->application_table_name,
    char *export_table->appaserver_select,
    char *export_table->appaserver_system_string,
    char *export_table->shell_filespecification,
    char *export_table->execute_bit_system_string,
    boolean output_prompt_boolean );

/* Process */
FILE *appaserver_input_pipe(
    appaserver_system_string );

FILE *appaserver_output_file(
    shell_filespecification );

fprintf(
    appaserver_output_file(),
    "%s\n",
    char *shell_script_open_string(
        application_name ) );

fprintf(
    appaserver_output_file(),
    "%s\n",
    char *shell_script_here_open_string(
        SHELL_SCRIPT_HERE_LABEL ) );

for char input[] in string_input(
    appaserver_input_pipe() ) )
{
    fprintf(
        appaserver_output_file(),
        "%s\n",
        char *insert_folder_string_sql_statement(
            SQL_DELIMITER,
            application_table_name,
            appaserver_select
                /* attribute_name_list_string */,
            input /* delimited_string */ ) );
}

pclose( appaserver_input_pipe() );

fprintf(
    appaserver_output_file(),
    "%s\n",
    char *shell_script_here_close_string(
        SHELL_SCRIPT_HERE_LABEL,
    0 /* not tee_appaserver_boolean */,
    0 /* not html_paragraph_wrapper_boolean */ ) );

fprintf(
    appaserver_output_file(),
    "%s\n",
    char *shell_script_close_string() );

fclose( appaserver_output_file() );

system( execute_bit_system_string );

if ( output_prompt_boolean )
{
    process_execution_count_increment(
        process_name );

    printf(
        "<h3>Created: %s</h3>\n",
        shell_filename );
}

/* Driver */
void export_table_spreadsheet_output(
    char *export_table->appaserver_system_string,
    char *export_table->spreadsheet_filespecification,
    char *export_table->
        appaserver_spreadsheet_heading_string,
    char *export_table->
        appaserver_spreadsheet_output_system_string,
    char *export_table->
        appaserver_link_anchor_html );

/* Process */
FILE *appaserver_output_file(
    spreadsheet_filespecification );

fprintf(
    appaserver_output_file(),
    "%s\n",
    appaserver_spreadsheet_heading_string );

fclose( appaserver_output_file() );

FILE *appaserver_output_pipe(
    appaserver_spreadsheet_output_system_string );

FILE *appaserver_input_pipe(
    appaserver_system_string );

for char input[] in string_input(
    appaserver_input_pipe() )
{
    fprintf(
        appaserver_output_pipe(),
        "%s\n",
        input );
}

pclose( appaserver_input_pipe() );
pclose( appaserver_output_pipe() );

if ( appaserver_link_anchor_html )
{
    process_execution_count_increment(
        process_name );

    printf(
        "%s\n",
        appaserver_link_anchor_html );
}

/* Driver */
void export_table_html_output(
    char *export_table->appaserver_system_string,
    char *export_table->html_table_system_string );

/* Process */
FILE *appaserver_output_pipe(
    html_table_system_string );

FILE *appaserver_input_pipe(
    appaserver_system_string );

for char input[] in string_input(
    appaserver_input_pipe() )
{
    fprintf(
        appaserver_output_pipe(),
        "%s\n",
        input );
}

pclose( appaserver_input_pipe );
pclose( appaserver_output_pipe );
```

EXPORT_TABLE (4)

```
/* Driver */
void export_table_insert_statement_output(
    char *export_table->
        appaserver_system_string,
    char *export_table->
        insert_folder_statement_system_string,
    char *output_pipe_string );

/* Process */
char *export_table_output_system_string(
    char *insert_folder_statement_system_string,
    char *output_pipe_string );

FILE *appaserver_output_pipe(
    export_table_output_system_string() );

FILE *appaserver_input_pipe(
    appaserver_system_string );

for char input[] in string_input(
    appaserver_input_pipe() )
{
    fprintf(
        appaserver_output_pipe(),
        “%s\n”,
        input );
}

pclose( appaserver_input_pipe );
pclose( appaserver_output_pipe );
```

SELECT_STATEMENT

```
/* Usage */
SELECT_STATEMENT *select_statement_fetch(
    char *select_statement_title );

/* Process */
char *select_statement_primary_where(
    char *select_statement_title );

char *appaserver_system_string(
    char *SELECT_STATEMENT_SELECT,
    char *SELECT_STATEMENT_TABLE,
    char *select_statement_primary_where() );

char *string_pipe_input(
    select_statement_system_string() );

if ( !string_pipe_input() )
{
    return (SELECT_STATEMENT *)0;
}

SELECT_STATEMENT *
    select_statement_parse(
        select_statement_title,
        string_pipe_input() );

/* Usage */
SELECT_STATEMENT *
    select_statement_parse(
        char *select_statement_title,
        char *input );

/* Process */
SELECT_STATEMENT *
    select_statement_calloc(
        void );

char *statement = strdup( input );

/* Attributes */
char *select_statement_title;
char *statement;
```

EXECUTE_SELECT (1)

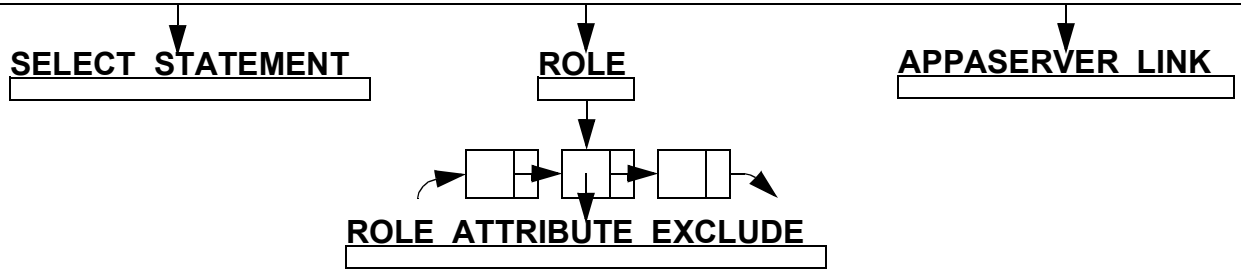
```
/* Usage */
EXECUTE_SELECT *execute_select_new(
    char *application_name,
    char *session_key,
    char *role_name,
    char *select_statement_title,
    char *statement,
    char *appaserver_parameter->data_directory );

/* Process */
EXECUTE_SELECT *execute_select_calloc(
    void );

if (    *select_statement_title
&&  strcmp(
          select_statement_title,
          "select_statement_title" ) != 0 )
{
     SELECT_STATEMENT *select_statement =
          select_statement_fetch(
                select_statement_title );

    if ( !select_statement ) exit( 1 );

    char *this->statement = select_statement->statement;
}
else
{
    this->statement =
        security_sql_injection_unescape(
            SECURITY_ESCAPE_CHARACTER_STRING,
            statement );

    this->statement =
        security_remove_semicolon(
            this->statement );
}

if (    !*this->statement
||      strcmp( this->statement, "statement" ) == 0 )
{
    error_message =
        "Please choose a select statement";

        return this;
}

ROLE *role =
    role_fetch(
        role_name,
        1 /* fetch_role_attribute_exclude_list */ );

LIST *column_name_list =
    LIST *execute_select_column_name_list(
        &error_message,
        role_name,
        role->role_attribute_exclude_list,
        this->statement );

if ( error_message
||  !list_length( column_name_list ) )
{
    return this;
}

boolean
    execute_select_first_column_asterisk_boolean(
        char *list_first( column_name_list )
            /* first_column_name */ );

if ( execute_select_first_column_asterisk_boolean() )
{
    LIST *column_name_list =
        LIST *execute_select_asterisk_name_list(
            role->role_attribute_exclude_list,
            list_first(
                execute_select_folder_name_list()
                    /* folder_name */ ) );

    char *this->statement =
        char *execute_select_asterisk_statement(
            this->statement,
            column_name_list );
}

LIST *execute_select_folder_name_list(
    &error_message,
    role_name,
    this->statement );

if ( error_message
||  !list_length( execute_select_folder_name_list() ) )
{
    return this;
}

char *execute_select_html_heading_string(
    column_name_list );

char *execute_select_html_system_string(
    SQL_DELIMITER,
    HTML_TABLE_QUEUE_TOP_BOTTOM,
    execute_select_html_heading_string() );
```

EXECUTE_SELECT (2)

```
/* Process (continued) */
APPASERVER_LINK *appaserver_link =
    appaserver_link_new(
        application_http_prefix(
            application_ssl_support_boolean(
                application_name ) ),
        application_server_address(),
        data_directory,
        “execute_select” /* filename_stem */,
        application_name,
        0 /* process_id */,
        session_key,
        (char *)0 /* begin_date_string */,
        (char *)0 /* end_date_string */,
        “csv” /* extension */ );

char *prompt_filename =
    appaserver_link->
        appaserver_link_prompt->
        filename;

char *output_filename =
    appaserver_link->
        appaserver_link_output->
        filename;

char *execute_select_spreadsheet_heading_string(
    SQL_DELIMITER,
    column_name_list );

char *execute_select_spreadsheet_system_string(
    SQL_DELIMITER,
    output_filename );

char *execute_select_error_filename(
    char *session_key );

char *execute_select_input_system_string(
    SQL_DELIMITER,
    this->statement,
    execute_select_error_filename() );

char *appaserver_link_anchor_html(
     prompt_filename,
     “_new” /* target_window */,
     “Link to spreadsheet” /* prompt_message */ );
```

EXECUTE_SELECT (4)

```
#define EXECUTE_BEGIN_STATE      0
#define EXECUTE_SELECT_STATE     1
#define EXECUTE_FROM_STATE       2

/* Attributes */
char *statement;
SELECT_STATEMENT *select_statement;
ROLE *role;
char *error_message;
LIST *column_name_list;
LIST *folder_name_list;
boolean first_column_asteric_boolean;
char *html_heading_string;
char *html_system_string;
APPASERVER_LINK *appaserver_link;
char *prompt_filename;
char *output_filename;
char *spreadsheet_heading_string;
char *spreadsheet_system_string;
char *error_filename;
char *input_system_string;
char *appaserver_link_anchor_html;
```



EXECUTE_SELECT (4)

```
/* Usage */
LIST *execute_select_column_name_list(
    char **error_message,
    char *role_name,
    LIST *role_attribute_exclude_list,
    char *statement );

/* Process */
int state = EXECUTE_BEGIN_STATE;
char column_name[];
int parenthesis_nest = 0;
char *token_ptr;
LIST *column_name_list = list_new();

*column_name = '\0';

while ( ( token_ptr = token_get( &statement ) ) )
{
    if ( *token_ptr == ';' )
    {
        break;
    }
    else
    if ( state == EXECUTE_BEGIN_STATE
    && strcasecmp( token_ptr, "select" ) != 0 )
    {
        *error_message = "Not a select statement";
        return (LIST *)0;
    }
    else
    if ( state == EXECUTE_BEGIN_STATE )
    {
        state = EXECUTE_SELECT_STATE;
    }
    else
    if (    strcasecmp( token_ptr, "from" ) == 0
    ||      strcasecmp( token_ptr, "where" ) == 0
    ||      strcasecmp( token_ptr, "order" ) == 0
    ||      strcasecmp( token_ptr, "group" ) == 0 )
    {
        list_set(
            column_name_list,
            strdup( column_name ) );

        break;
    }
    else
    if ( *token_ptr == '(' )
    {
        strcat( column_name, token_ptr );
        parenthesis_nest++;
    }
    else
    if ( *token_ptr == ')' )
    {
        if ( !parenthesis_nest )
        {
            *error_message = "Mismatched ')'";
            return (LIST *)0;
        }

        strcat( column_name, token_ptr );
        parenthesis_nest--;
    }
    else
    if ( *token_ptr == ',' && !parenthesis_nest )
    {
        list_set(
            column_name_list,
            strdup( column_name ) );

        *column_name = '\0';
    }
    else
    {
        boolean execute_select_exists_attribute_exclude(
            role_attribute_exclude_list,
            token_ptr /* attribute_name */ );

        if ( execute_select_exists_attribute_exclude() )
        {
            char *execute_select_exclude_violation(
                role_name,
                token_ptr /* attribute_name */ );

            *error_message =
                execute_select_exclude_violation();

            return (LIST *)0;
        }

        strcat( column_name, token_ptr );
    }
}

if ( state == EXECUTE_BEGIN_STATE )
    return (LIST *)0;
else
    return column_name_list;
```

EXECUTE_SELECT (5)

```
/* Usage */
char *execute_select_asterisk_statement(
    char *statement,
    LIST *column_name_list );

/* Process */
char *list_display_delimited(
    column_name_list, ‘,’ );

char *string_strcpy(
    buffer[],
    statement,
    sizeof ( buffer ) );

char *string_search_replace(
    string_strcpy() /* source_destination */,
    “*” /* search_string */,
    list_display_delimited() /* replace_string */ );

char *strdup( string_search_replace() );

/* Usage */
LIST *execute_select_asterisk_name_list(
    LIST *role_attribute_exclude_list,
    char *folder_name );

/* Process */
LIST *role_attribute_exclude_lookup_name_list(
    APPASERVER_LOOKUP_STATE,
    APPASERVER_UPDATE_STATE,
    role_attribute_exclude_list );

LIST *folder_attribute_list(
    folder_name,
    role_attribute_exclude_lookup_name_list()
        /* exclude_attribute_name_list */,
    0 /* not fetch_attribute */,
    0 /* not cache_boolean */ );

LIST *folder_attribute_name_list(
    (char *)0 /* folder_name */,
    folder_attribute_list() );
}

/* Usage */
char *execute_select_spreadsheet_heading_string(
    const char SQL_DELIMITER,
    LIST *column_name_list );

/* Process */
list_display_delimited(
    column_name_list,
    SQL_DELIMITER );
```

EXECUTE_SELECT (6)

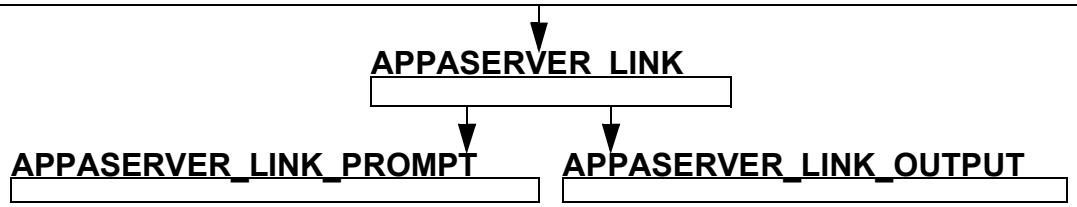
```
/* Usage */
LIST *execute_select_folder_name_list(
    char **error_message,
    char *role_name,
    char *statement );

/* Process */
int state = EXECUTE_BEGIN_STATE;
char *token_ptr;
LIST *folder_name_list = list_new();

while ( ( token_ptr = token_get( &statement ) ) )
{
    if ( *token_ptr == ';' )
    {
        break;
    }
    else
    if ( strcasecmp( token_ptr, "from" ) == 0 )
    {
        state = EXECUTE_FROM_STATE;
    }
    else
    if ( state == EXECUTE_FROM_STATE )
    {
        if (    strcasecmp( token_ptr, "where" ) == 0
        ||      strcasecmp( token_ptr, "order" ) == 0
        ||      strcasecmp( token_ptr, "group" ) == 0 )
        {
            break;
        }

        if ( *token_ptr == ',' ) continue;

        boolean execute_select_exists_role_folder(
            ROLE_SYSTEM,
            role_name,
            token_ptr /* folder_name */ );

        if ( !execute_select_exists_role_folder() )
        {
            char *execute_select_not_exists_role_violation(
                role_name,
                token_ptr /* folder_name */ );

            *error_message =
                execute_select_not_exists_role_violation();
            return (LIST *)0;
        }

        list_set(
            folder_name_list,
            strdup( token_ptr )
                /* folder_name */ );
    }
}

if ( state == EXECUTE_BEGIN_STATE )
    return (LIST *)0;
else
    return folder_name_list;

/* Usage */
boolean execute_select_exists_attribute_exclude(
    LIST *role_attribute_exclude_list,
    char *attribute_name );

/* Process */
boolean role_attribute_exclude_exists_permission(
    APPASERVER_LOOKUP_STATE,
    attribute_name,
    role_attribute_exclude_list );

if ( role_attribute_exclude_exists_permission() )
{
    return 1;
}

boolean role_attribute_exclude_exists_permission(
    APPASERVER_UPDATE_STATE,
    attribute_name,
    role_attribute_exclude_list );

if ( role_attribute_exclude_exists_permission() )
{
    return 1;
}

return 0;

/* Usage */
char *execute_select_exclude_violation(
    char *role_name,
    char *attribute_name );

/* Process */
sprintf(    static violation[],
            "%s: %s, logged in as role: %s",
            EXECUTE_SELECT_ATTRIBUTE_VIOLATION,
            attribute_name,
            role_name );

/* Usage */
boolean execute_select_exists_role_folder(
    const char *ROLE_SYSTEM,
    char *role_name,
    char *folder_name );

/* Process */
static LIST *lookup_name_list = {0};

int strcmp(
    role_name,
    ROLE_SYSTEM );

if ( strcmp() == 0 ) return 1;

if ( !lookup_name_list )
{
    LIST *role_folder_lookup_list(
        role_name );

    LIST *lookup_name_list =
        role_folder_name_list(
            role_folder_lookup_list() );
}

return
boolean list_exists_string(
    folder_name,
    lookup_name_list );

/* Usage */
char *execute_select_not_exists_role_violation(
    char *role_name,
    char *folder_name );

/* Process */
sprintf(    static violation[],
            "%s: %s, logged in as role: %s",
            EXECUTE_SELECT_FOLDER_VIOLATION,
            folder_name,
            role_name );
```

EXECUTE_SELECT (7)

```
/* Driver */
/* Returns row_count */
int execute_select_generate(
    char *execute_select->html_system_string,
    char *execute_select->output_filename,
    char *execute_select->spreadsheet_heading_string,
    char *execute_select->spreadsheet_system_string,
    char *execute_select->error_filename,
    char *execute_select->input_system_string );

/* Process */
FILE *execute_select_html_output_pipe(
    char *html_system_string );

timlib_zap_file( output_filename );

FILE *execute_select_spreadsheet_output_pipe(
    char *spreadsheet_system_string );

fprintf(    execute_select_spreadsheet_output_pipe(),
            "%s\n",
            spreadsheet_heading_string );

FILE *execute_select_input_pipe(
    char *input_system_string );

int row_count = 0;

for char input[] in
    string_input( execute_select_input_pipe() )
{
    fprintf(
        execute_select_html_output_pipe(),
        "%s\n",
        input );

    fprintf(
        execute_select_spreadsheet_output_pipe(),
        "%s\n",
        input );

    row_count++;
}

pclose( execute_select_input_pipe() );
pclose( execute_select_spreadsheet_output_pipe() );
pclose( execute_select_html_output_pipe() );

char *string_file_fetch(
    error_filename,
    (char *)0 /* delimiter */ );

if ( string_file_fetch() )
{
    printf(
        "<h3>%s</h3>\n",
        string_file_fetch() );
}

timlib_remove_file( error_filename );

/* Driver */
printf(   "<h3>Row count: %d</h3>\n",
          row_count );

printf(   "%s\n",
          execute_select->
              appaserver_link_anchor_html );

process_execution_count_increment(
    process_name );

/* Usage */
char *execute_select_html_system_string(
    char SQL_DELIMITER,
    int HTML_TABLE_QUEUE_TOP_BOTTOM,
    char *execute_select_html_heading_string() );

/* Process */
snprintf(   char system_string[],
            sizeof ( system_string ),
            "queue_top_bottom_lines.e %d |"
            "html_table.e '' \"%s\" '%c'",
            HTML_TABLE_QUEUE_TOP_BOTTOM,
            execute_select_html_heading_string,
            SQL_DELIMITER );

/* Usage */
char *execute_select_spreadsheet_system_string(
    char SQL_DELIMITER,
    char *output_filename );

/* Process */
snprintf(
    char system_string[],
    sizeof ( system_string ),
    "double_quote_comma_delimited.e '%c' >> %s",
    SQL_DELIMITER,
    output_filename );

/* Usage */
char *execute_select_input_system_string(
    const char SQL_DELIMITER,
    char *statement,
    char *execute_select_error_filename() );

/* Process */
snprintf(   char system_string[],
            sizeof ( system_string ),
            "echo \"%s\" |"
            "grep -v '^#' |"
            "sql_quick.e '%c' 2>%s",
            statement,
            SQL_DELIMITER,
            execute_select_error_filename );

/* Usage */
char *execute_select_html_heading_string(
    LIST *column_name_list );

/* Process */
strcpy(
    char heading_string[],
    list_display_delimited(
        column_name_list, '%' ) );

string_search_replace(
    heading_string /* source_destination */,
    "," /* search_character */,
    "\\," /* replace_character */ );

string_search_replace_character(
    heading_string /* source_destination */,
    '%' /* search_character */,
    ',' /* replace_character */ );
```

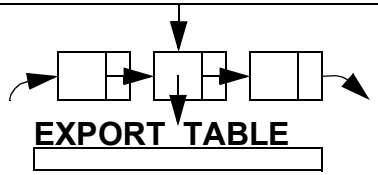
PKZIP

```
/* Usage */
PKZIP *pkzip_new(
    char *application_name,
    LIST *filename_list,
    char *filename_stem,
    char *appaserver_parameter->data_directory );

/* Process */
PKZIP *pkzip_calloc( void );

char *date_now_yyyy_mm_dd(
    date_utc_offset() );

APPASERVER_LINK *appaserver_link =
    appaserver_link_new(
        application_http_prefix(
            application_ssl_support_boolean(
                application_name ) ),
        application_server_address(),
        data_directory,
        filename_stem,
        application_name,
        getpid() /* process_id */,
        (char *)0 /* session_key */,
        date_now_yyyy_mm_dd() /* begin_date_string */,
        (char *)0 /* end_date_string */,
        "zip" /* extension */ );

char *pkzip_system_string(
    PKZIP_EXECUTABLE,
    filename_list,
    appaserver_link->output->filename );

char *appaserver_link_anchor_html(
    appaserver_link->
        appaserver_link_prompt->
        filename,
    "_new" /* target_window */,
    "Link to zip file" /* prompt_message */ );

/* Usage */
char *pkzip_system_string(
    const char *PKZIP_EXECUTABLE,
    LIST *filename_list,
    char *output_filename );

/* Process */
char system_string[];
char *ptr = system_string;

ptr += sprintf(
        ptr,
        "%s %s",
        PKZIP_EXECUTABLE,
        output_filename );

for char *filename in filename_list
{
    ptr += sprintf(
        ptr,
        " %s",
        filename );
}

/* Attributes */
char *date_now_yyyy_mm_dd;
APPASERVER_LINK *appaserver_link;
char *system_string;
char *appaserver_link_anchor_html;

/* Driver */
system( pkzip->system_string );

printf( "%s\n", pkzip->appaserver_link_anchor_html );
```



EXPORT_APPLICATION (1)

```
/* Usage */
EXPORT_APPLICATION *export_application_new(
    char *application_name,
    boolean system_folders_boolean,
    char *appaserver_parameter->data_directory );

/* Process */
EXPORT_APPLICATION *export_application_calloc(
    void );

char *export_application_where(
    ROLE_FOLDER_TABLE,
    ROLE_SYSTEM,
    ROLE_SUPERVISOR,
    FOLDER_PRIMARY_KEY,
    system_folders_boolean );

LIST *export_application_folder_name_list(
    FOLDER_PRIMARY_KEY,
    FOLDER_TABLE,
    export_application_where() );

if ( !list_rewind( export_application_folder_name_list() )
{
    return this;
}

LIST *export_table_list = list_new();

for char *folder_name in
    export_application_folder_name_list()
{
    list_set(
        export_table_list,
        export_table_new(
            application_name,
            folder_name,
            data_directory,
            (char *)0 /* where */ ) );
}

/* Usage */
LIST *export_application_folder_name_list(
    const char *FOLDER_PRIMARY_KEY,
    const char *FOLDER_TABLE,
    char *export_application_where() );

/* Process */
char *appaserver_system_string(
    FOLDER_PRIMARY_KEY /* select */,
    FOLDER_TABLE,
    export_application_where );

return
list_pipe_fetch(
    appaserver_system_string() );

/* Usage */
char *export_application_where(
    const char *ROLE_FOLDER_TABLE,
    const char *ROLE_SYSTEM,
    const char *ROLE_SUPERVISOR,
    const char *FOLDER_PRIMARY_KEY,
    boolean system_folders_boolean );

/* Process */
char *export_application_role_folder_subquery(
    ROLE_FOLDER_TABLE,
    ROLE_SYSTEM,
    ROLE_SUPERVISOR,
    FOLDER_PRIMARY_KEY,
    system_folders_boolean );

snprintf(
    static char where[],
    sizeof ( where ),
    "ifnull(exclude_application_export_yn,'n' ) != 'y' "
    "and %s "
    "and %s != 'session'"
    "and %s != 'null'",
    export_application_role_folder_subquery(),
    FOLDER_PRIMARY_KEY,
    FOLDER_PRIMARY_KEY );

/* Usage */
char *export_application_role_folder_subquery(
    const char *ROLE_FOLDER_TABLE,
    const char *ROLE_SYSTEM,
    const char *FOLDER_PRIMARY_KEY,
    boolean system_folders_boolean );

/* Process */
char *export_application_role(
    const char *ROLE_SYSTEM,
    const char *ROLE_SUPERVISOR,
    boolean system_folders_boolean );

snprintf(
    static char subquery[],
    sizeof ( subquery ),
    "%s in (select distinct %s from %s where "
    "role = '%s')",
    FOLDER_PRIMARY_KEY.
    FOLDER_PRIMARY_KEY,
    ROLE_FOLDER_TABLE,
    export_application_role() );

/* Attributes */
char *where;
LIST *folder_name_list;
LIST *export_table_list;
```



EXPORT_APPLICATION (2)

```
/* Driver */
void export_application_spreadsheet_output(
    char *application_name,
    LIST *export_application->export_table_list,
    char *appaserver_parameter->data_directory );

/* Process */
for EXPORT_TABLE *export_table in export_table_list
{
    void export_table_spreadsheet_output(
        export_table->appaserver_system_string,
        export_table->spreadsheet_filename,
        export_table->
          appaserver_spreadsheet_heading_string,
        export_table->
          appaserver_spreadsheet_output_system_string,
        (char *)0 /* appaserver_link_anchor_html */ );
}

LIST *export_table_spreadsheet_filename_list(
    export_table_list );

PKZIP *pkzip =
    pkzip_new(
        application_name,
        export_table_spreadsheet_filename_list(),
        "export" /* filename_stem */,
        data_directory );

system( pkzip->system_string );

printf( "%s\n", pkzip->appaserver_link_anchor_html );

process_execution_count_increment(
    process_name );

/* Driver */
void export_application_shell_output(
    char *application_name,
    LIST *export_application->export_table_list,
    char *appaserver_parameter->data_directory );

/* Process */
for EXPORT_TABLE *export_table in export_table_list
{
    void export_table_shell_output(
        (char *)0 /* application_name -- no if statement */,
        export_table->application_table_name,
        export_table->appaserver_select,
        export_table->appaserver_system_string,
        export_table->shell_filename,
        export_table->execute_bit_system_string,
        0 /* not output_prompt_boolean );
}

LIST *export_table_shell_filename_list(
    export_table_list );

PKZIP *pkzip =
    pkzip_new(
        application_name,
        export_table_shell_filename_list(),
        "export" /* filename_stem */,
        data_directory );

system( pkzip->system_string );

printf( "%s\n", pkzip->appaserver_link_anchor_html );

process_execution_count_increment(
    process_name );

/* Driver */
void export_application_html_output(
    LIST *export_application->export_table_list );

/* Process */
for EXPORT_TABLE *export_table in export_table_list
{
    void export_table_html_output(
        export_table->appaserver_system_string,
        export_table->html_table_system_string );
}
```

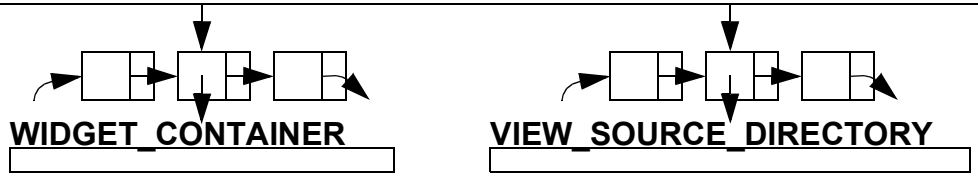
VIEW_SOURCE (1)

```
/* Usage */
VIEW_SOURCE *view_source_new(
    char *application_name,
    char *process_name,
    char *appaserver_parameter->mount_point,
    enum view_source_enum view_source_enum );

/* Process */
VIEW_SOURCE *view_source_calloc( void );

char *document_process_title_string(
    char *process_name );

char *document_process_sub_title_string();

char *document_body_html(
    (char *)0 /* menu_html */,
    document_process_title_string(),
    document_process_sub_title_string(),
    (char *)0 /* sub_sub_title_string */,
    (char *)0 /* body_onload_open_tag */,
    (char *)0 /* document_body_hide_preload_html */,
    (char *)0 /* notepad_html */ );

LIST *widget_container_list = list_new();

LIST *view_source_directory_list = list_new();

LIST *application_relative_source_directory_list(
    application_name );

for char *relative_source_directory_name
    in application_relative_source_directory_list()
{
    VIEW_SOURCE_DIRECTORY *
        view_source_directory =
            view_source_directory_new(
                mount_point,
                view_source_enum,
                relative_directory_name );

    if ( view_source_directory )
    {
        list_set(
            view_source_directory_list,
            view_source_directory );

        list_set_list(
            widget_container_list,
            view_source_directory->widget_container_list );
    }
}

boolean view_source_directory_boolean(
    view_source_directory_list,
    "src_appaserver" );

if ( !view_source_directory_boolean() )
{
    VIEW_SOURCE_DIRECTORY *
        view_source_directory =
            view_source_directory_new(
                appaserver_parameter_mount_point,
                view_source_diagrams,
                "src_appaserver" );

    if ( view_source_directory )
    {
        list_set(
            view_source_directory_list,
            view_source_directory );

        list_set_list(
            widget_container_list,
            view_source_directory->widget_container_list );
    }
}

boolean view_source_directory_boolean(
    view_source_directory_list,
    "template" );

if ( !view_source_directory_boolean() )
{
    VIEW_SOURCE_DIRECTORY *
        view_source_directory =
            view_source_directory_new(
                mount_point,
                view_source_enum,
                "template" );

    if ( view_source_directory )
    {
        list_set(
            view_source_directory_list,
            view_source_directory );

        list_set_list(
            widget_container_list,
            view_source_directory->widget_container_list );
    }
}

boolean view_source_directory_boolean(
    view_source_directory_list,
    "utility" );

if ( !view_source_directory_boolean() )
{
    VIEW_SOURCE_DIRECTORY *
        view_source_directory =
            view_source_directory_new(
                mount_point,
                view_source_enum,
                "utility" );

    if ( view_source_directory )
    {
        list_set(
            view_source_directory_list,
            view_source_directory );

        list_set_list(
            widget_container_list,
            view_source_directory->widget_container_list );
    }
}
```

VIEW_SOURCE (2)

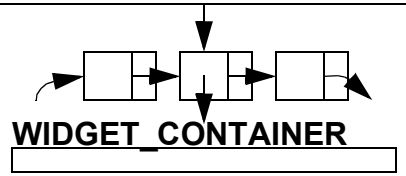
```
/* Process (continued) */
boolean view_source_directory_boolean(
    view_source_directory_list,
    "library" );

if ( !view_source_directory_boolean() )
{
    VIEW_SOURCE_DIRECTORY *
        view_source_directory =
            view_source_directory_new(
                mount_point,
                view_source_enum,
                "library" );

    if ( view_source_directory )
    {
        list_set(
            view_source_directory_list,
            view_source_directory );

        list_set_list(
            widget_container_list,
            view_source_directory->widget_container_list );
    }
}

list_set(
    widget_container_list,
    (widget_container =
        widget_container_new(
            button, (char *)0 ) ) );

widget_container->widget_button->button =
    button_to_top();

char *widget_container_list_html(
    (char *)0 /* state */,
    -1 /* row_number to omit index suffix */,
    (char *)0 /* background_color */,
    widget_container_list );

char *document_form_html(
    (char *)0 /* document_html */,
    (char *)0 /* document_head_html */,
    (char *)0 /* document_head_close_tag */,
    document_body_html(),
        widget_container_list_html()
            /* form_html */,
        document_body_close_tag(),
        document_close_tag() );

/* Driver */
printf(
    "%s\n",
    view_source->document_form_html );

process_execute_increment(
    process_name );

/* Attributes */
enum view_source_enum view_source_enum;
char *document_process_title_string;
char *document_body_html;
LIST *widget_container_list;
LIST *application_relative_source_directory_list;
LIST *view_source_directory_list;
char *widget_container_list_html;
char *document_form_html;
```

```
enum view_source_enum = {
        view_source_code,
        view_source_diagrams }
```



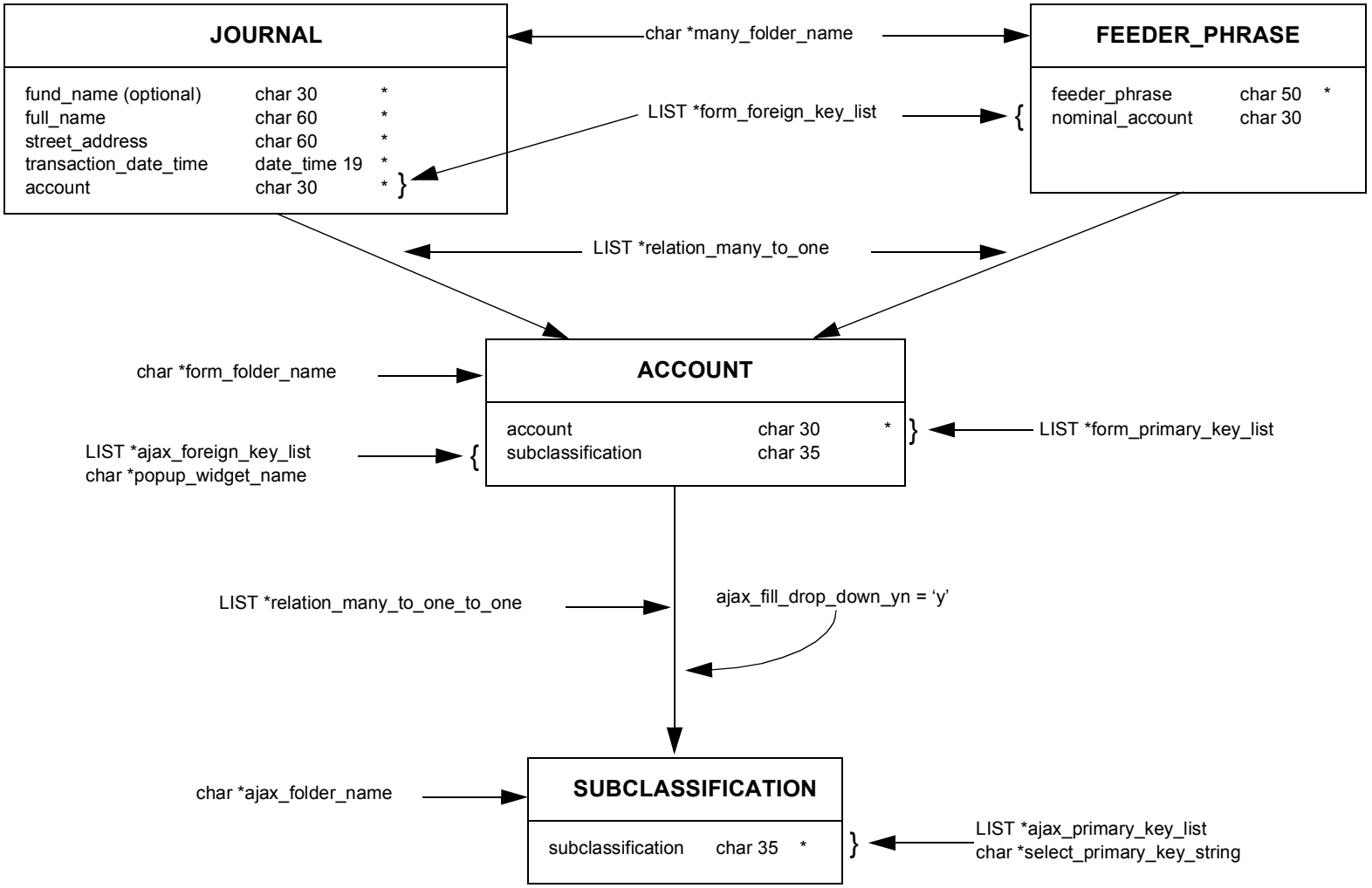
VIEW_SOURCE_DIRECTORY

```
/* Usage */
VIEW_SOURCE_DIRECTORY *
    view_source_directory_new(
        char *appaserver_parameter_mount_point,
        enum view_source_enum view_source_enum,
        char *relative_directory_name );

/* Process */
VIEW_SOURCE_DIRECTORY *
    view_source_directory_calloc(
        void );

char *view_source_directory_absolute_name(
    char *mount_point,
    char *relative_directory_name );

int chdir( view_source_directory_absolute_name() );

if ( chdir() != 0 ) exit( 1 );

char *view_source_directory_ls_system_string(
    view_source_enum );

LIST *view_source_directory_filename_list(
    char *view_source_directory_ls_system_string );

LIST *view_source_directory_widget_container_list(
    relative_directory_name,
    view_source_directory_filename_list() );

/* Usage */
char *view_source_directory_ls_system_string(
    enum view_source_enum view_source_enum );

/* Process */
if ( view_source_enum == view_source_code )
{
    return
"ls -1 makefile *.[ch] *.sh *.js *.awk"
" 2>/dev/null | sort";
}
else
if ( view_source_enum == view_source_diagrams )
{
    return
"ls -1 *.xls *.jpg *.doc *.pdf *.png *.txt *.ppt *.xml"
" 2>/dev/null | sort";
}

/* Usage */
LIST *view_source_directory_widget_container_list(
    char *relative_directory_name,
    LIST *view_source_directory_filename_list() );

/* Process */
LIST *widget_container_list = list_new();

char *view_source_directory_heading_string(
    char *relative_directory_name );

list_set(
    widget_container_list,
    widget_container_new(
        non_edit_text,
        view_source_directory_heading_string() ) );

list_set(
    widget_container_list,
    widget_container_new(
        unordered_list_open, (char *)0 ) );

for char *filename
    in view_source_directory_filename_list
{
    list_set(
        widget_container_list,
        WIDGET_CONTAINER *widget_container_new(
            list_item, (char *)0 ) );

    char *view_source_directory_href_string(
        char *relative_directory_name,
        char *filename );

    list_set(
        widget_container_list,
        WIDGET_CONTAINER *
            view_source_directory_widget_container(
                filename,
                view_source_directory_href_string() ) );
}

list_set(
    widget_container_list,
    widget_container_new(
        unordered_list_close, (char *)0 ) );

/* Usage */
WIDGET_CONTAINER *
    view_source_directory_widget_container(
        char *filename,
        char *view_source_directory_href_string );

/* Process */
WIDGET_CONTAINER *widget_container =
    widget_container_new(
        anchor, (char *)0 );

widget_container->anchor->hypertext_reference =
    view_source_directory_href_string;

widget_container->anchor->target_frame =
     "_new";

widget_container->anchor->prompt =
     filename;

/* Usage */
boolean view_source_directory_boolean(
    LIST *view_source_directory_list,
    const char *relative_directory_name );

/* Process */

/* Attributes */
char *relative_directory_name;
char *absolute_name;
LIST *filename_list;
LIST *widget_container_list;
```



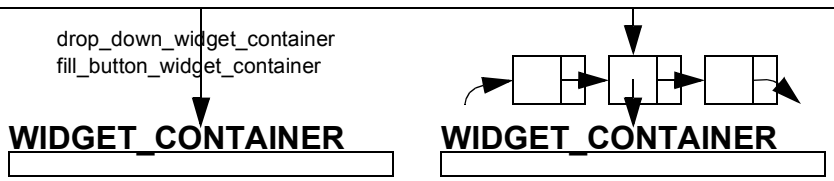
WIDGET_CONTAINER

AJAX_CLIENT (1)



AJAX_CLIENT (2)

```
/* Usage */
AJAX_CLIENT *ajax_client_new(
    char *many_folder_name,
    char *form_folder_name,
    LIST *form_foreign_key_list,
    LIST *form_primary_key_list,
    char *ajax_folder_name,
    LIST *ajax_primary_key_list,
    LIST *ajax_foreign_key_list,
    boolean top_select_boolean );

/* Process */
AJAX_CLIENT *ajax_client_calloc(
    void );

char *ajax_client_form_widget_name(
    const char SQL_DELIMITER,
    LIST *form_foreign_key_list );

WIDGET_CONTAINER *
    ajax_client_drop_down_widget_container(
        many_folder_name,
        form_folder_name,
        form_foreign_key_list,
        top_select_boolean,
        ajax_client_form_widget_name() );

WIDGET_CONTAINER *
    fill_button_widget_container =
        widget_container_new(
            button,
            AJAX_CLIENT_FILL_LABEL
                /* widget_name */ );

char *ajax_client_form_widget_id(
    const char SQL_DELIMITER,
    char *many_folder_name,
    LIST *form_foreign_key_list );

char *ajax_client_fill_button_action_string(
    ajax_folder_name,
    ajax_client_form_widget_id() );

fill_button_widget_container->
    widget_button->
    button->
    action_string =
        ajax_client_fill_button_action_string();

char *ajax_client_select_primary_key_string(
    ajax_primary_key_list );

char *ajax_client_popup_widget_name(
    const char SQL_DELIMITER,
    LIST *ajax_foreign_key_list );

LIST *widget_container_list = list_new();

list_set(
    widget_container_list,
    drop_down_widget_container );

list_set(
    widget_container_list,
    fill_button_widget_container );

/* Usage */
char *ajax_client_fill_button_action_string(
    char *ajax_folder_name,
    char *ajax_client_form_widget_id() );

/* Process */
snprintf(
    char action_string[],
    sizeof ( action_string ),
    "fork_%s_ajax_window( '%s_$row' )",
    ajax_folder_name,
    ajax_client_form_widget_id );

/* Usage */
char *ajax_client_select_primary_key_string(
    LIST *ajax_primary_key_list );

/* Process */
char *list_delimited(
    ajax_primary_key_list,
    ',' );

/* Attributes */
char *many_folder_name;
char *form_folder_name;
char *ajax_folder_name;
char *form_widget_name;
char *form_widget_id;
char *fill_button_action_string;
WIDGET_CONTAINER *drop_down_widget_container;
WIDGET_CONTAINER *fill_button_widget_container;
char *select_primary_key_string;
char *popup_widget_name;
LIST *widget_container_list;
```



AJAX_CLIENT (3)

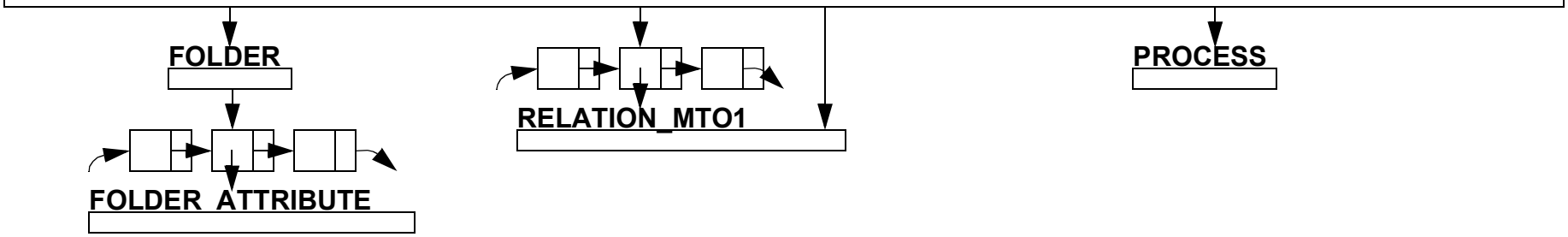
```
/* Usage */
char *ajax_client_list_javascript(
    char *session_key,
    char *login_name,
    char *role_name,
    LIST *ajax_client_list );

/* Process */
char javascript[];
char *ptr = javascript;

for AJAX_CLIENT *ajax_client in ajax_client_list
{
    char *ajax_client_execute_window_system_string(
        AJAX_CLIENT_WINDOW_EXECUTABLE,
        AJAX_SERVER_POST_EXECUTABLE,
        session_key,
        login_name,
        role_name,
        ajax_client->form_folder_name,
        ajax_client->ajax_folder_name,
        ajax_client->popup_widget_name,
        ajax_client->select_primary_key_string );

    SPOOL *spool_new(
        ajax_client_execute_window_system_string(),
        0 /* not capture_stderr_boolean */ );

    pclose( spool_new()->output_pipe );

    LIST *spool_list(
        spool_new()->output_filespecification,
        0 /* not capture_stderr_boolean */ );

    free( ajax_client_execute_window_system_string() );

    for char *spool_line in spool_list()
    {
        ptr += sprintf(
            ptr,
            "%s\n",
            spool_line );
    }

    if ( spool_list() ) list_free( spool_list() );
}

/* Usage */
char *ajax_client_execute_window_system_string(
    const char *AJAX_CLIENT_WINDOW_EXECUTABLE,
    const char *AJAX_SERVER_POST_EXECUTABLE,
    char *session_key,
    char *login_name,
    char *role_name,
    char *form_folder_name,
    char *ajax_folder_name,
    char *popup_widget_name,
    char *select_primary_key_string );

/* Process */
snprintf(
    char system_string[],
    sizeof ( system_string ),
    "%s %s %s %s %s %s %s %s '%s'",
    AJAX_CLIENT_WINDOW_EXECUTABLE,
    AJAX_SERVER_POST_EXECUTABLE,
    session_key,
    login_name,
    role_name,
    form_folder_name,
    ajax_folder_name,
    popup_widget_name,
    select_primary_key_string );

/* Usage */
AJAX_CLIENT *ajax_client_relation_mto1_new(
    RELATION_MTO1 *relation_mto1,
    LIST *relation_mto1_to_one_list,
    boolean top_select_boolean );

/* Process */
if ( !relation_mto1->
        relation->
        ajax_fill_drop_down )
{
    return NULL;
}

RELATION_MTO1 *relation_mto1_ajax =
    relation_mto1_ajax_fill_seek(
        relation_mto1_to_one_list );

if ( !relation_mto1_ajax ) return NULL;

AJAX_CLIENT *ajax_client_new(
    relation_mto1->
        many_folder_name,
    relation_mto1->
        one_folder_name
        /* form_folder_name */,
    relation_mto1->
        relation_foreign_key_list
        /* form_foreign_key_list */,
    relation_mto1->
        one_folder->
        folder_attribute_primary_key_list
        /* form_primary_key_list */,
    relation_mto1_ajax->
        one_folder_name
        /* ajax_folder_name */,
    relation_mto1_ajax->
        one_folder->
        folder_attribute_primary_key_list,
        /* ajax_primary_key_list */,
    relation_mto1_ajax->
        relation_foreign_key_list
        /* ajax_foreign_key_list */,
    top_select_boolean );
```

AJAX_CLIENT (4)

```
/* Usage */
WIDGET_CONTAINER *
    ajax_client_drop_down_widget_container(
        char *many_folder_name,
        char *form_folder_name,
        LIST *form_foreign_key_list,
        boolean top_select_boolean,
        char *ajax_client_form_widget_name );

/* Process */
WIDGET_CONTAINER *
    drop_down_widget_container =
        widget_container_new(
            drop_down,
            ajax_client_form_widget_name );

drop_down_widget_container->
    drop_down->
    foreign_key_list =
        form_foreign_key_list;

drop_down_widget_container->
    drop_down->
    top_select_boolean =
        top_select_boolean;

drop_down_widget_container->
    drop_down->
    many_folder_name =
        many_folder_name;

drop_down_widget_container->
    heading_string =
        form_folder_name;

drop_down_widget_container->
    drop_down->
    widget_drop_down_option_list =
        list_new();

WIDGET_DROP_DOWN_OPTION *
    widget_drop_down_option_new(
        QUERY_IS_NULL );

list_set(
    drop_down_widget_container->
        drop_down->
        widget_drop_down_option_list,
    widget_drop_down_option_new() );

WIDGET_DROP_DOWN_OPTION *
    widget_drop_down_option_new(
        QUERY_NOT_NULL );

list_set(
    drop_down_widget_container->
        drop_down->
        widget_drop_down_option_list,
    widget_drop_down_option_new() );
```

AJAX_SERVER (1)

```
/* Usage */
AJAX_SERVER *ajax_server_new(
    char *session_folder->session->appliction_name,
    char *session_folder->folder_name
        /* form_folder_name */,
    char *session_folder->role_name,
    char *security_sql_injection_escape(
        ajax_folder_name ),
    char *security_sql_injection_escape(
        value ) );

/* Process */
AJAX_SERVER *ajax_server_calloc(
    void );

FOLDER *form_folder =
    folder_fetch(
        form_folder_name,
        (LIST *)0 /* exclude_attribute_name_list */,
        /* Sets folder_attribute_primary_list */
        /* Sets folder_attribute_primary_key_list */
        /* Sets folder_attribute_name_list */
        1 /* fetch_folder_attribute_list */,
        0 /* not fetch_attribute */,
        1 /* cache_boolean */ );

LIST *relation_mto1_list(
    form_folder_name
        /* many_folder_name */,
    form_folder->folder_attribute_primary_key_list
        /* many_folder_primary_key_list */ );

RELATION_MTO1 *relation_mto1_to_one =
    relation_mto1_seek(
        ajax_folder_name,
        relation_mto1_list() );

if ( !relation_mto1_to_one ) exit( 1 );

if ( form_folder->populate_drop_down_process_name )
{
    PROCESS *populate_drop_down_process =
        process_fetch(
            form_folder->
                populate_drop_down_process_name,
            (char *)0 /* document_root */,
            (char *)0 /* relative_source_directory */,
            1 /* check_executable_inside_filesystem */,
            appaserver_parameter_mount_point() );
}
```



AJAX_SERVER (2)

```
/* Process (continued) */
if ( populate_drop_down_process )
{
    char *ajax_server_foreign_key_list_string(
        char SQL_DELIMITER,
        LIST *relation_mto1_to_one->
            relation_foreign_key_list );

    char *system_string =
        char *ajax_server_process_system_string(
            value,
            populate_drop_down_process,
            ajax_server_foreign_key_list_string() );
}
else
{
    LIST *ajax_server_attribute_data_list(
        const char SQL_DELIMITER,
        char *value );

    char *query_data_where(
        (char *)0 /* appaserver_table_name */,
        relation_mto1_to_one->relation_foreign_key_list
            /* where_attribute_name_list */,
        ajax_server_attribute_data_list()
            /* where_attribute_data_list */,
        form_folder->folder_attribute_list );

    char *ajax_server_select_string(
        LIST *form_folder->
            folder_attribute_primary_key_list );

    char *system_string =
        char *ajax_server_folder_system_string(
            application_name,
            form_folder_name,
            query_data_where(),
            ajax_server_select_string() );
}

LIST *ajax_server_result_list( system_string );

if ( list_length( ajax_server_result_list() ) )
{
    char *ajax_server_return_string(
        SQL_DELIMITER,
        WIDGET_DROP_DOWN_DASH_DELIMITER,
        AJAX_SERVER_DATA_DELIMITER
            /* probably "@&" */,
        AJAX_SERVER_FORMAT_DELIMITER
            /* probably '|' */,
        ajax_server_result_list() );
}

/* Attributes */
FOLDER *form_folder;
LIST *relation_mto1_list;
RELATION_MTO1 *relation_mto1_to_one;
PROCESS *populate_drop_down_process;
char *foreign_key_list_string;
LIST *attribute_data_list;
char *query_data_where;
char *select_string;
char *system_string;
LIST *result_list;
char *return_string;

/* Driver */
if ( !ajax_server->return_string )
    printf( "\n" );
else
    printf( "%s\n", ajax_server->return_string );
```

AJAX_SERVER (3)

```
/* Usage */
char *ajax_server_process_system_string(
    char *value,
    PROCESS *populate_drop_down_process,
    char *ajax_server_foreign_key_list_string() );

/* Process */
DICTIONARY *dictionary_small();

dictionary_set(
    dictionary_small(),
    ajax_server_foreign_key_list_string /* key */,
    value /* datum */ );

char *query_drop_down_process_command_line(
    (char *)0 /* application_name */,
    (char *)0 /* session_key */,
    (char *)0 /* login_name */,
    (char *)0 /* role_name */,
    (char *)0 /* state */,
    (char *)0 /* many_folder_name */,
    (char *)0 /* one_folder_name */,
    (char *)0 /* related_column */,
    populate_drop_down_process->process_name,
    dictionary_small(),
    (char *)0 /* query_drop_down_where_string */,
    populate_drop_down_process->command_line,
    (char *)0 /* appaserver_error_filespecification */ );

/* Usage */
char *ajax_server_folder_system_string(
    char *application_name,
    char *form_folder_name,
    char *query_data_where(),
    char *ajax_server_select_string() );

/* Process */
snprintf(
    system_string[],
    sizeof ( system_string ),
    "select     application=%s "
    "           select=%s "
    "           folder=%s "
    "           where=\"%s\" "
    "           order=select",
    application_name,
    ajax_server_select_string,
    form_folder_name,
    query_data_where );

/* Usage */
LIST *ajax_server_result_list(
    char *system_string );

/* Process */
SPOOL *spool_new(
    system_string,
    0 /* not capture_stderr_boolean */ );

pclose( spool_new()->output_pipe );

LIST *spool_list( spool_new()->output_filespecification );

/* Usage */
char *ajax_server_return_string(
    const char SQL_DELIMITER,
    const char *
        WIDGET_DROP_DOWN_DASH_DELIMITER,
    const char *
        AJAX_SERVER_DATA_DELIMITER
        /* probably "@&" */,
    const char AJAX_SERVER_FORMAT_DELIMITER
        /* probably '|' */,
    LIST *ajax_server_result_list() );

/* Process */
char *list_first( ajax_server_result_list );

boolean string_exists_character(
    list_first(),
    AJAX_SERVER_FORMAT_DELIMITER );

if ( string_exists_character() )
{
    char *data_string =
        char *ajax_server_delimited_data_string(
            AJAX_SERVER_DATA_DELIMITER,
            AJAX_SERVER_FORMAT_DELIMITER,
            ajax_server_result_list );

    char *format_string =
        char *ajax_server_delimited_format_string(
            AJAX_SERVER_DATA_DELIMITER,
            AJAX_SERVER_FORMAT_DELIMITER,
            ajax_server_result_list );
}
else
{
    char *data_string =
        char *ajax_server_data_string(
            AJAX_SERVER_DATA_DELIMITER,
            ajax_server_result_list );

    char *format_string =
        char *ajax_server_list_format_string(
            SQL_DELIMITER,
            AJAX_SERVER_DATA_DELIMITER,
            WIDGET_DROP_DOWN_DASH_DELIMITER,
            ajax_server_result_list );
}

char *ajax_server_format_return_string(
    AJAX_SERVER_FORMAT_DELIMITER,
    data_string,
    format_string );

free( data_string );
free( format_string );
```

AJAX_SERVER (4)

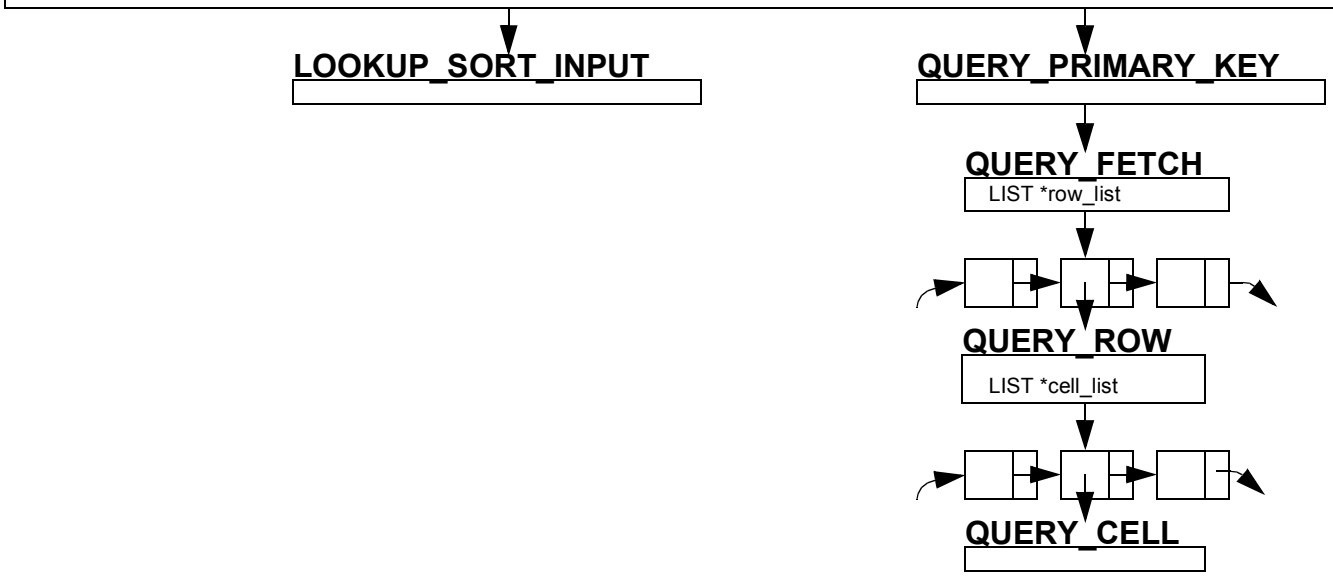
```
/* Usage */
char *ajax_server_delimited_data_string(
    const char *AJAX_SERVER_DATA_DELIMITER
        /* probably "@&" */,
    const char AJAX_SERVER_FORMAT_DELIMITER
        /* probably '|' */,
    LIST *ajax_server_result_list() );

/* Process */
char data_string[];
char *ptr = data_string;

for char *result in ajax_server_result_list
{
    if ( !list_first_boolean( ajax_server_result_list ) )
    {
        ptr += sprintf( ptr,
            "%s",
            AJAX_SERVER_DATA_DELIMITER );
    }

    piece(
        piece_buffer,
        AJAX_SERVER_FORMAT_DELIMITER,
        result,
        0 );

    ptr += sprintf( ptr,
        "%s",
        piece_buffer );
}

/* Usage */
char *ajax_server_delimited_format_string(
    const char AJAX_SERVER_DATA_DELIMITER
        /* probably "@&" */
    const char AJAX_SERVER_FORMAT_DELIMITER
        /* probably '|' */,
    LIST *ajax_server_result_list() );

/* Process */
char format_string[];
char *ptr = format_string;

for char *result in ajax_server_result_list
{
    if ( !list_first_boolean( ajax_server_result_list ) )
    {
        ptr += sprintf( ptr,
            "%s",
            AJAX_SERVER_DATA_DELIMITER );
    }

    piece(
        piece_buffer,
        AJAX_SERVER_FORMAT_DELIMITER,
        result,
        1 );

    ptr += sprintf( ptr,
        "%s",
        piece_buffer );
}

/* Usage */
char *ajax_server_data_string(
    const char *AJAX_SERVER_DATA_DELIMITER
        /* probably "@&",
    LIST *ajax_server_result_list );

/* Process */
char data_string[];
char *ptr = data_string;

for char *result in ajax_server_result_list
{
    if ( !list_first_boolean( ajax_server_result_list ) )
    {
        ptr += sprintf( ptr,
            "%s",
            AJAX_SERVER_DATA_DELIMITER );
    }

    ptr += sprintf( ptr, "%s", result );
}

/* Usage */
char *ajax_server_list_format_string(
    const char SQL_DELIMITER,
    const char AJAX_SERVER_DATA_DELIMITER
        /* probably "@&" */,
    const *WIDGET_DROP_DOWN_DASH_DELIMITER,
    LIST *ajax_server_result_list() );

/* Process */
char format_string[];
char *ptr = format_string;

char *sql_delimiter_string( SQL_DELIMITER );

for char *result in ajax_server_result_list
{
    if ( !list_first_boolean( ajax_server_result_list ) )
    {
        ptr += sprintf( ptr,
            "%s",
            AJAX_SERVER_DATA_DELIMITER );
    }

    char *ajax_server_format_string(
        WIDGET_DROP_DOWN_DASH_DELIMITER,
        sql_delimiter_string(),
        result );

    ptr += sprintf( ptr,
        "%s",
        ajax_server_format_string() );
}
```

AJAX_SERVER (5)

```
/* Usage */
char *ajax_server_format_string(
    const *WIDGET_DROP_DOWN_DASH_DELIMITER,
    char *sql_delimter_string(),
    char *result );

/* Process */
char *string_initial_capital(
    static format_string[],
    result );

char *string_search_replace(
    format_string /* source_destination */,
    sql_delimiter_string, /* search_string */,
    WIDGET_DROP_DOWN_DASH_DELIMITER
        /* replace_string */ );

/* Usage */
char *ajax_server_format_return_string(
    const char AJAX_SERVER_FORMAT_DELIMITER
        /* probably '|' */,
    char *data_string,
    char *format_string );

/* Process */
sprintf(
    format_return_string[],
    "%s%c%s",
    data_string,
    AJAX_SERVER_FORMAT_DELIMITER,
    format_string );
```

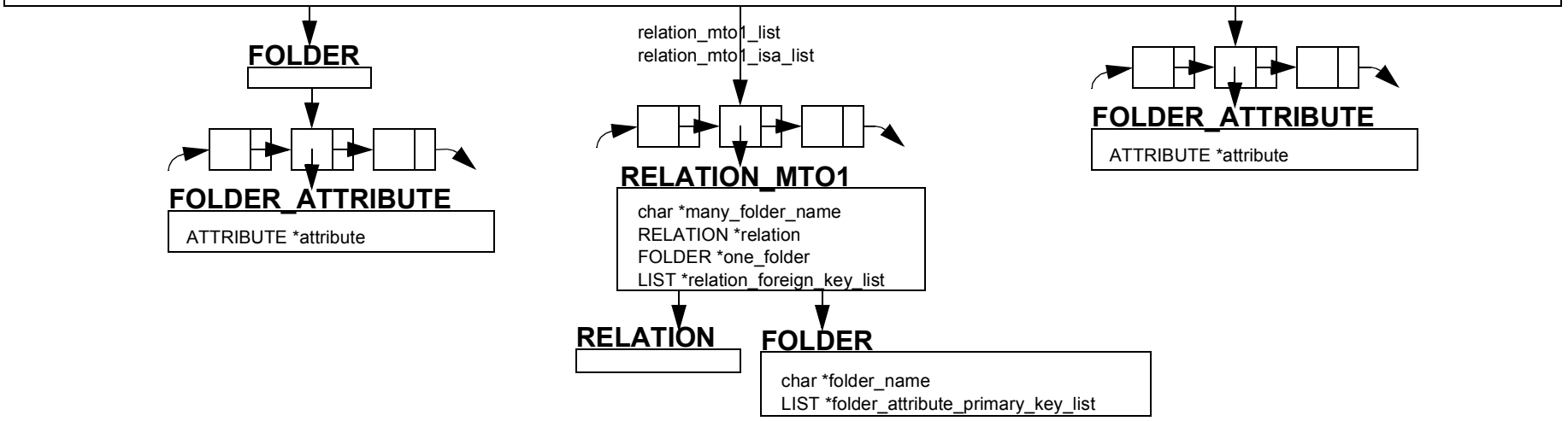
LOOKUP_SORT

```
/* Usage */
LOOKUP_SORT *lookup_sort_new(
    char *session_folder->session->application_name,
    char *session_folder->session->login_name,
    char *session_folder->role_name,
    char *session_folder->folder_name,
    DICTIONARY *post_dictionary->
       original_post_dictionary );

/* Process */
LOOKUP_SORT *lookup_sort_calloc( void );

LOOKUP_SORT_INPUT *lookup_sort_input =
    lookup_sort_input_new(
       application_name,
       login_name,
       role_name,
       folder_name,
       original_post_dictionary );

QUERY_PRIMARY_KEY *query_primary_key =
   /* Note: frees each where. */
   query_primary_key_fetch(
       application_name,
       login_name,
       folder_name,
       lookup_sort_input->security_entity_where,
       lookup_sort_input->relation_mto1_list,
       lookup_sort_input->relation_mto1_isa_list,
       lookup_sort_input->
          folder_attribute_append_isa_list,
       lookup_sort_input->
          dictionary_separate->
          non_prefixed_dictionary
          /* query_dictionary */,
       lookup_sort_input->
          attribute_name
          /* additional_attribute_name */ );

int lookup_sort_row_count(
    list_length(
          query_primary_key->
             query_fetch->
             row_list )
             /* row_list_length */ );

/* Attributes */
LOOKUP_SORT_INPUT *lookup_sort_input;
QUERY_PRIMARY_KEY *query_primary_key;
int row_count;
```



LOOKUP_SORT_INPUT (1)

```
/* Usage */
LOOKUP_SORT_INPUT *lookup_sort_input_new(
    char *application_name,
    char *login_name,
    char *role_name,
    char *folder_name,
    DICTIONARY *original_post_dictionary );

/* Process */
LOOKUP_SORT_INPUT *lookup_sort_input_calloc( void );

FOLDER *folder =
    folder_fetch(
        folder_name,
        (LIST *)0 /* exclude_attribute_name_list */,
        /* Sets folder_attribute_primary_list */
        /* Sets folder_attribute_primary_key_list */
        /* Sets folder_attribute_name_list */
        1 /* fetch_folder_attribute_list */,
        1 /* fetch_attribute */,
        1 /* cache_boolean */ );

LIST *relation_mto1_list(
    folder_name
        /* many_folder_name */,
    folder->folder_attribute_primary_key_list
        /* many_folder_primary_key_list */ );

LIST *relation_mto1_isa_list(
    (LIST *)0 /* mto1_isa_list Pass in null */,
    folder_name
        /* many_folder_name */,
    folder->folder_attribute_primary_key_list
        /* many_primary_key_list */,
    0 /* not fetch_relation_one2m_list */,
    0 /* not fetch_relation_mto1_list */ );

LIST *folder_attribute_append_isa_list(
    folder->folder_attribute_list /* append_isa_list */,
    relation_mto1_isa_list() );

LIST *folder_attribute_name_list(
    (char *)0 /* folder_name */,
    folder_attribute_append_isa_list() );

LIST *folder_attribute_date_name_list(
    folder_attribute_append_isa_list() );
```



LOOKUP_SORT_INPUT (2)

```
/* Process (continued) */
DICTIONARY_SEPARATE_POST_PROMPT_LOOKUP *
   dictionary_separate =
      dictionary_separate_post_prompt_lookup_new(
         original_post_dictionary,
         application_name,
         login_name,
         relation_mto1_list(),
         folder_attribute_name_list(),
         folder_attribute_date_name_list(),
         folder_attribute_append_isa_list() );

ROLE *role =
   role_fetch(
      role_name,
      0 /* not fetch_role_attribute_exclude_list */ );

FOLDER_ROW_LEVEL_RESTRICTION *
   folder_row_level_restriction =
      folder_row_level_restriction_fetch(
         folder_name );

SECURITY_ENTITY *security_entity =
   security_entity_new(
      login_name,
      folder_name,
      folder_row_level_restriction->
         non_owner_forbid ||
      folder_row_level_restriction->
         non_owner_viewonly,
      role->override_row_restrictions );

char *lookup_sort_input_attribute_name(
   char *ATTRIBUTE_NAME_SORT_ORDER,
   char *ATTRIBUTE_NAME_DISPLAY_ORDER,
   char *ATTRIBUTE_NAME_SEQUENCE_NUMBER,
      LIST *folder->folder_attribute_name_list );

/* Attributes */
FOLDER *folder;
LIST *relation_mto1_list;
LIST *relation_mto1_isa_list;
LIST *folder_attribute_append_isa_list;
LIST *folder_attribute_name_list;
LIST *folder_attribute_date_name_list;
DICTIONARY_SEPARATE_POST_PROMPT_LOOKUP *
   dictionary_separate;
ROLE *role;
FOLDER_ROW_LEVEL_RESTRICTION *
   folder_row_level_restriction;
SECURITY_ENTITY *security_entity;
char *attribute_name;
```

DICTIONARY_SEPARATE_POST_PROMPT_LOOKUP

```
DICTIONARY *drillthru_dictionary
DICTIONARY *prompt_dictionary
DICTIONARY *no_display_dictionary
DICTIONARY *non_prefixed_dictionary
```

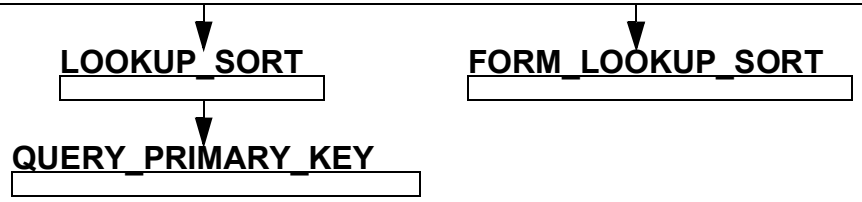
ROLE

SECURITY_ENTITY

FOLDER_ROW_LEVEL_RESTRICTION

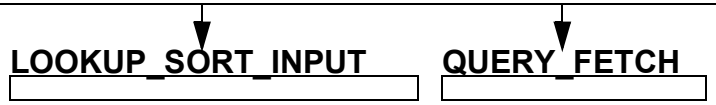
LOOKUP_SORT_FORM

```
/* Usage */
LOOKUP_SORT_FORM *lookup_sort_form_new(
    char *LOOKUP_SORT_EXECUTABLE,
    char *session_folder->session->application_name,
    char *session_folder->session->session_key,
    char *session_folder->session->login_name,
    char *session_folder->role_name,
    char *session_folder->folder_name,
    DICTIONARY *post_dictionary->
        original_post_dictionary );

/* Process */
LOOKUP_SORT_FORM *lookup_sort_form_calloc(
    void );

LOOKUP_SORT *lookup_sort =
    lookup_sort_new(
        application_name,
        login_name,
        role_name,
        folder_name,
        original_post_dictionary );

if ( lookup_sort->row_count == 0 )
    return this;

LIST *lookup_sort_form_javascript_filename_list(
    void );

char *lookup_sort_form_action_string(
    char *LOOKUP_SORT_EXECUTABLE,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *folder_name );

char *query_dictionary_hidden_html =
    dictionary_separate_hidden_html(
        DICTIONARY_SEPARATE_QUERY_PREFIX,
        lookup_sort->
            dictionary_separate->
        query_dictionary );

FORM_LOOKUP_SORT *form_lookup_sort =
    form_lookup_sort_new(
        lookup_sort->
            lookup_sort_input->
            folder->
            folder_attribute_primary_key_list,
        lookup_sort->lookup_sort_input->attribute_name,
        lookup_sort->
            query_primary_key->
            query_fetch->
            row_list,
        lookup_sort_form_action_string() );

/* Driver */
void document_process_output(
    application_name,
    lookup_sort_form->javascript_filename_list,
    "sort" /* title_string */ );

if ( !lookup_sort_form->lookup_sort->row_count )
{
    printf(   "<h3>%s</h3>\n",
              LOOKUP_SORT_FORM_NO_ROWS_MESSAGE );
}
else
{
    printf(
        "%s\n",
        lookup_sort_form->
            form_lookup_sort->
            form_html );
}

void document_close();

/* Attributes */
LOOKUP_SORT *lookup_sort;
LIST *javascript_filename_list;
char *action_string;
char *query_dictionary_hidden_html;
FORM_LOOKUP_SORT *form_lookup_sort;
```



LOOKUP_SORT_EXECUTE (1)

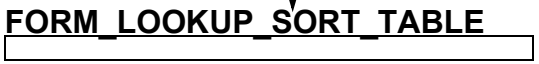
```
/* Usage */
LOOKUP_SORT_EXECUTE *lookup_sort_execute_new(
    char *session_folder->session->application_name,
    char *session_folder->session->login_name,
    char *session_folder->role_name,
    char *session_folder->folder_name,
    DICTIONARY *post_dictionary->
        original_post_dictionary,
    char *appaserver_error_filename() );

/* Process */
LOOKUP_SORT_EXECUTE *lookup_sort_execute_calloc(
    void );

LOOKUP_SORT_INPUT *lookup_sort_input =
    lookup_sort_input_new(
        application_name,
        login_name,
        role_name,
        folder_name,
        original_post_dictionary );

QUERY_FETCH *lookup_sort_execute_query_fetch(
    lookup_sort_input->
        folder->
        folder_attribute_primary_key_list,
    lookup_sort_input->
        attribute_name,
    original_post_dictionary );

char *appaserver_table_name(
    folder_name );

LIST *lookup_sort_execute_update_statement_list(
    lookup_sort->
        lookup_sort_input->
        folder->
        folder_attribute_primary_key_list,
    lookup_sort->
        lookup_sort_input->
        attribute_name,
        lookup_sort_execute->
        query_fetch->
        row_list,
    appaserver_table_name() );

char *lookup_sort_execute_system_string(
    appaserver_error_filename );

/* Usage */
LIST *lookup_sort_execute_update_statement_list(
    LIST *primary_key_list,
    char *lookup_sort_input_attribute_name(),
    LIST *query_fetch_row_list,
    char *appaserver_table_name );

/* Process */
LIST *list = list_new();

for QUERY_ROW *query_row in
    query_fetch_row_list
{
    list_set(
        list,
        char *lookup_sort_execute_update_statement(
            primary_key_list,
            lookup_sort_input_attribute_name,
            appaserver_table_name,
            query_row->cell_list ) );
}

/* Usage */
char *lookup_sort_execute_update_statement(
    LIST *primary_key_list,
    char *lookup_sort_input_attribute_name,
    char *appaserver_table_name,
    LIST *query_row_cell_list );

/* Process */
char update_statement[];
char *ptr = update_statement;

QUERY_CELL *query_cell =
    list_last(
        query_row_cell_list );

ptr +=
    sprintf(
        ptr,
        "update %s set %s = %s where ",
        appaserver_table_name,
        lookup_sort_input_attribute_name,
        query_cell->select_datum );

list_rewind( query_row_cell_list );

for char *primary_key in primary_key_list
{
    if ( !list_at_first( primary_key_list ) )
        ptr += sprintf( ptr, " and " );

    query_cell = list_get( query_row_cell_list );

    ptr +=
        sprintf(
            ptr,
            "%s = '%s'",
            primary_key,
            query_cell->select_datum );

    list_next( query_row_cell_list );
}

ptr += sprintf( ptr, ";" );

/* Attributes */
LOOKUP_SORT_INPUT *lookup_sort_input;
QUERY_FETCH *query_fetch;
char *appaserver_table_name;
LIST *update_statement_list;
char *system_string;
```

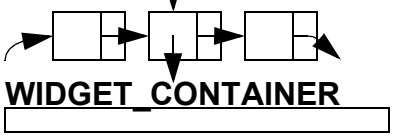


```
/* Usage */
QUERY_FETCH *lookup_sort_execute_query_fetch(
    LIST *folder_attribute_primary_key_list,
    char *attribute_name,
    DICTIONARY *original_post_dictionary );

/* Process */
int dictionary_highest_index(
    original_post_dictionary );

QUERY_FETCH *query_fetch = query_fetch_calloc();

query_fetch->row_list = list_new();

for int index = 1 to dictionary_highest_index()
{
    QUERY_ROW *query_row =
        lookup_sort_execute_query_row(
            folder_attribute_primary_key_list,
            attribute_name,
            original_post_dictionary,
            index );

    list_set(
        query_fetch->row_list,
        query_row );
}

/* Usage */
QUERY_ROW *lookup_sort_execute_query_row(
    LIST *folder_attribute_primary_key_list,
    char *attribute_name,
    DICTIONARY *original_post_dictionary,
    int index );

/* Process */
QUERY_ROW *query_row = query_row_calloc();

query_row->cell_list = list_new();

for char *primary_key in
    folder_attribute_primary_key_list
{
    QUERY_CELL *query_cell =
        lookup_sort_execute_query_cell(
            primary_key /* attribute_name */,
            original_post_dictionary,
            index );

    list_set(
        query_row->cell_list,
        query_cell );
}

QUERY_CELL *query_cell =
    lookup_sort_execute_query_cell(
        attribute_name,
        original_post_dictionary,
        index );

list_set(
    query_row->cell_list,
    query_cell );

/* Usage */
QUERY_CELL *lookup_sort_execute_query_cell(
    char *attribute_name,
    DICTIONARY *original_post_dictionary,
    int index );

/* Process */
QUERY_CELL *query_cell = query_cell_calloc();

char *dictionary_index_get(
    attribute_name /* key */,
    original_post_dictionary,
    index );

query_cell->select_datum = dictionary_index_get();

/* Usage */
char *lookup_sort_execute_system_string(
    char *appaserver_error_filename );

/* Process */
snprintf(
    static system_string[],
    sizeof ( system_string ),
    "tee_appaserver.sh |"
    "sql.e 2>>%s",
    appaserver_error_filename );

/* Usage */
void lookup_sort_execute_update(
    LIST *lookup_sort_execute->
        update_statement_list,
    char *lookup_sort_execute->
        system_string );

/* Process */
FILE *appaserver_output_pipe(
    system_string );

for char *update_statement in
    update_statement_list
{
    fprintf(
        appaserver_output_pipe(),
        "%s\n",
        update_statement );
}
pclose( appaserver_output_pipe() );

/* Driver */
void document_process_output(
    application_name,
    (LIST *)0 /* javascript_filename_list */,
    "sort" /* title_string */ );

void lookup_sort_execute_update(
    lookup_sort_execute->
        update_statement_list );

printf(
    "<h3>%s</h3>\n",
    LOOKUP_SORT_EXECUTE_COMPLETE_MESSAGE );

void document_close();
```

FORM_LOOKUP_SORT

```
/* Usage */
FORM_LOOKUP_SORT *form_lookup_sort_new(
    LIST *folder_attribute_primary_key_list,
    char *lookup_sort_input_attribute_name(),
    LIST *query_fetch_row_list,
    char *lookup_sort_form_action_string() );

/* Process */
FORM_LOOKUP_SORT *form_lookup_sort_calloc(
    void );

LIST *widget_container_list = list_new();

char *form_lookup_sort_renumber_onclick(
    lookup_sort_input_attribute_name );

LIST *form_lookup_sort_button_container_list(
    form_lookup_sort_renumber_onclick() );

list_set_list(
    widget_container_list,
    form_lookup_sort_button_container_list() );

FORM_LOOKUP_SORT_TABLE *
    form_lookup_sort_table =
        form_lookup_sort_table_new(
            folder_attribute_primary_key_list,
            lookup_sort_input_attribute_name,
            query_fetch_row_list );

list_set_list(
    widget_container_list,
    form_lookup_sort_table->
        widget_container_list );

char *widget_container_list_html(
    (char *)0 /* state */,
    -1 /* row_number to omit index suffix */,
    (char *)0 /* background_color */,
    widget_container_list );

char *form_tag(
    FORM_LOOKUP_SORT_NAME,
    lookup_sort_form_action_string,
    FRAMESET_TABLE_FRAME /* target_frame */ );

char *form_html(
    form_tag(),
    widget_container_list_html(),
    (char *)0 /* hidden_html */,
    form_close_tag() );

/* Usage */
LIST *form_lookup_sort_button_container_list(
   char *form_lookup_sort_renumber_onclick() );

/* Process */
LIST *widget_button_submit_reset_container_list(
    void );

LIST *widet_container_list =
   widget_button_submit_reset_container_list();

list_set(
   widget_container_list,
   ( WIDGET_CONTAINER *widget_container =
      widget_container_new(
         button, "Renumber" ) ) );

widget_container->
   widget_button->
   button->
   action_string =
      form_lookup_sort_renumber_onclick;

/* Usage */
char *form_lookup_sort_renumber_onclick(
   char *lookup_sort_input_attribute_name );

/* Process */
snprintf(
   static char onclick[],
   sizeof ( onclick ),
   "sort_order_renumber( '%s' )",
   lookup_sort_attribute_name );

/* Attributes */
char *renumber_onclick;
LIST *button_container_list;
LIST *widget_container_list;
FORM_LOOKUP_SORT_TABLE *form_lookup_sort_table;
char *widget_container_list_html /* freed */;
char *form_tag;
char *form_html;
```





FORM_LOOKUP_SORT_TABLE (1)

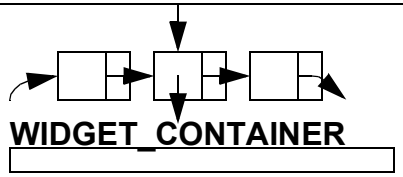
```
/* Usage */
FORM_LOOKUP_SORT_TABLE *
    form_lookup_sort_table_new(
        LIST *folder_attribute_primary_key_list,
        char *lookup_sort_input_attribute_name(),
        LIST *query_fetch_row_list );

/* Process */
FORM_LOOKUP_SORT_TABLE *
    form_lookup_sort_table_calloc( void );

LIST *widget_container_list = list_new();

list_set(
    widget_container_list,
    ( WIDGET_CONTAINER *widget_container =
        widget_container_new(
            table_open, (char *)0 ) ) );

widget_container->table_open->border_boolean = 1;

LIST *form_lookup_sort_table_heading_container_list(
    folder_attribute_primary_key_list,
    lookup_sort_input_attribute_name );

list_set_list(
    widget_container_list,
    form_lookup_sort_table_heading_container_list() );

char *form_lookup_sort_table_onclick(
    primary_key_list );

int row_number = 1;

for QUERY_ROW *query_row in query_fetch_row_list
{
    list_set(
        widget_container_list,
        widget_container_new(
            table_row, (char *)0 ) );

    list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );

    list_set(
        widget_container_list,
        ( widget_container =
            widget_container_new(
                widget_radio, “move” ) ) );

    char *form_lookup_sort_table_label(
        row_number,
        “move” /* prefix */ );

    widget_container->radio->value =
        form_lookup_sort_table_label();

    widget_container->radio->prompt = “Move”;

    list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );

    list_set(
        widget_container_list,
        ( widget_container =
            widget_container_new(
                widget_radio, “here” ) ) );

    char *form_lookup_sort_table_label(
        row_number,
        “here” /* prefix */ );

    widget_container->radio->value =
        form_lookup_sort_table_label();

    widget_container->radio->prompt = “Here”;

    widget_container->radio->onclick =
        form_lookup_sort_table_onclick();
```

FORM_LOOKUP_SORT_TABLE (2)

```
/* Process (continued) */
    list_rewind( query_row_cell_list );

    for char *primary_key in
        folder_attribute_primary_key_list
    {
        list_set(
            widget_container_list,
            widget_container_new(
                table_data, (char *)0 ) );

        char *form_lookup_sort_table_label(
            row_number,
            primary_key /* prefix */ );

        list_set(
            widget_container_list,
            ( widget_container =
                widget_container_new(
                    character,
                    form_lookup_sort_table_label() ) ) );

        widget_container->
            text->
            attribute_width_max_length =
                FORM_LOOKUP_SORT_TEXT_WIDTH;

        widget_container->
            text->
            widget_text_display_size =
                FORM_LOOKUP_SORT_TEXT_DISPLAY_SIZE;

        QUERY_CELL *query_cell =
            list_get(
                query_row_cell_list );

        widget_container->text->value_string =
                query_cell->select_datum;

        widget_container->text->vewonly = 1;

        list_next( query_row_cell_list );
    }

    list_set(
        widget_container_list,
        widget_container_new(
            table_data, (char *)0 ) );

    char *form_lookup_sort_table_label(
        row_number,
        lookup_sort_input_attribute_name
            /* prefix */ );

    list_set(
        widget_container_list,
        ( widget_container =
            widget_container_new(
                text,
                form_lookup_sort_table_label() ) ) );

        widget_container->
            text->
            attribute_width_max_length =
                FORM_LOOKUP_SORT_TEXT_DISPLAY_SIZE;

        widget_container->
            text->
            widget_text_display_size =
                FORM_LOOKUP_SORT_ORDER_DISPLAY_SIZE;

        QUERY_CELL *query_cell =
            list_get(
                query_row_cell_list );

        widget_container->
            text->
            value_string =
                query_cell->select_datum;

        widget_container->text->viewonly = 1;

        row_number++;
    }

    list_set(
        widget_container_list,
        widget_container_new(
            table_close, (char *)0 ) );
```

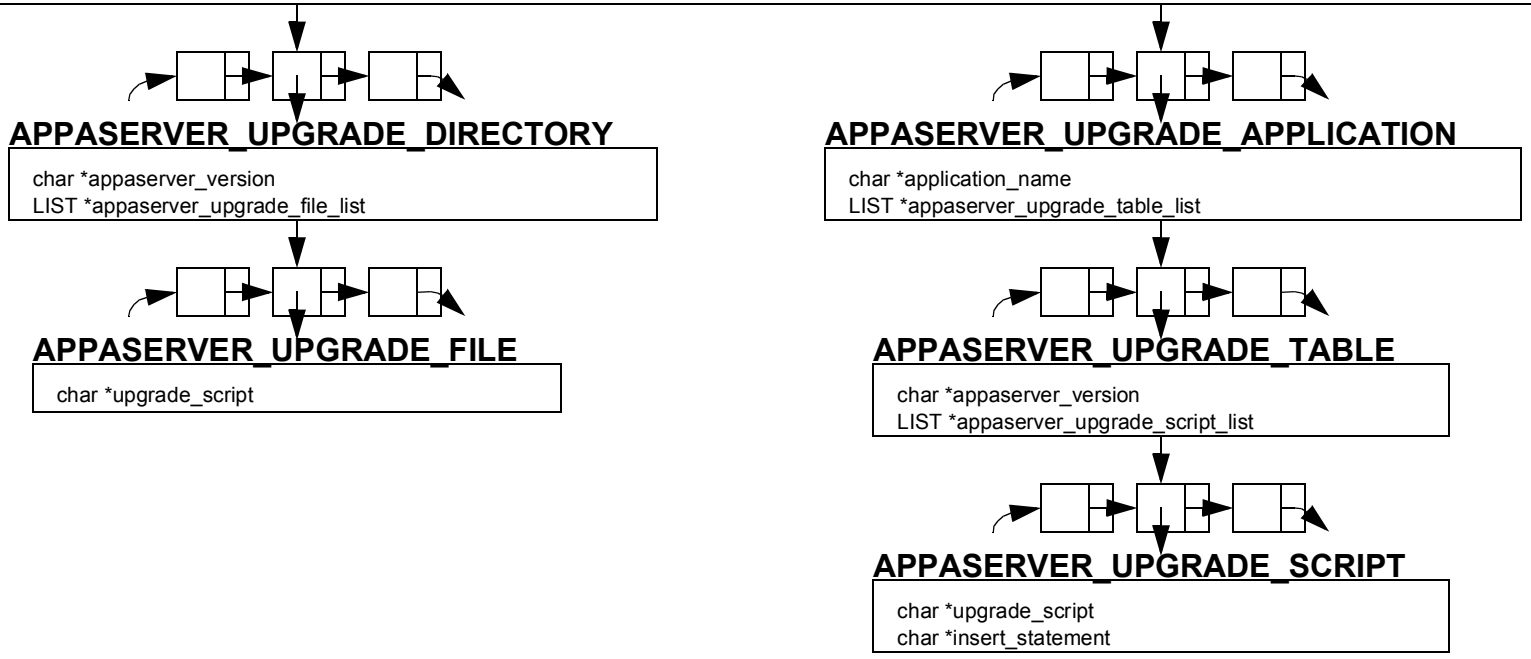
FORM_LOOKUP_SORT_TABLE (3)

```
/* Usage */
char *form_lookup_sort_table_label(
    int row_number,
    char *prefix );

/* Process */
snprintf(
    char label[],
    sizeof ( label ),
    "%s_%d",
    prefix,
    row_number );

/* Usage */
LIST *form_lookup_sort_table_heading_container_list(
    LIST *folder_attribute_primary_key_list,
    char *lookup_sort_input_attribute_name );

/* Process */
LIST *widget_container_list = list_new();

list_set(
    widget_container_list,
    widget_container_new(
        table_row, (char *)0 ) );

list_set(
    widget_container_list,
    widget_container_new(
        table_heading, "Move" ) );

list_set(
    widget_container_list,
    widget_container_new(
        table_heading, "Here" ) );

for char *primary_key in
    folder_attribute_primary_key_list
{
    char *form_lookup_sort_table_primary_heading(
        primary_key );

    list_set(
        widget_container_list,
        widget_container_new(
            table_heading,
            form_lookup_sort_table_primary_heading() ) );
}

char *form_lookup_sort_table_attribute_heading(
    lookup_sort_input_attribute_name );

list_set(
    widget_container_list,
    widget_container_new(
        table_heading,
        form_lookup_sort_table_attribute_heading() ) );

/* Usage */
char *form_lookup_sort_table_primary_heading(
    char *primary_key );

/* Process */
sprintf(    char heading[],
            "*%s",
            string_initial_capital(
                destination[],
                primary_key ) );

/* Usage */
char *form_lookup_sort_table_attribute_heading(
    char *lookup_sort_input_attribute_name );

/* Process */
string_initial_capital(
    static char heading[],
    lookup_sort_input_attribute_name );

/* Usage */
char *form_lookup_sort_table_onclick(
    LIST *primary_key_list );

/* Process */
char *primary_key_list_string =
    list_delimited(
        primary_key_list,
        ',' );

snprintf(
    static char onclick[],
    sizeof ( onclick ),
    "sort_order_move( '%s' )",
    primary_key_list_string );

/* Attributes */
LIST *widget_container_list;
LIST *heading_container_list;
char *onclick;
```



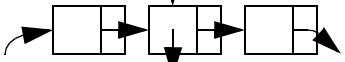
APPASERVER_UPGRADE (1)

```
/* Usage */
APPASERVER_UPGRADE *appaserver_upgrade_fetch(
    char *APPLICATION_LOG_EXTENSION,
    int argc,
    char **argv,
    char *application_name,
    boolean step_boolean,
    char *appaserver_parameter->log_directory,
    char *appaserver_parameter->mount_point );

/* Process */
APPASERVER_UPGRADE *appaserver_upgrade_calloc(
    void );

if (    !application_name
&&  !step_boolean )
{
    LIST *appaserver_upgrade_application_list_argv(
        int argc,
        char **argv );
}

LIST *appaserver_upgrade_application_name_list(
    APPLICATION_LOG_EXTENSION,
    application_name,
    log_directory,
    appaserver_upgrade_application_list_argv() );

if ( !list_length(
        appaserver_upgrade_application_name_list() )
{
    return this;
}

char *appaserver_upgrade_root(
    mount_point );

LIST *appaserver_upgrade_directory_list(
    appaserver_upgrade_root() );

if ( list_length( appaserver_upgrade_directory_list() )
{
    LIST *appaserver_upgrade_version_list(
        LIST *appaserver_upgrade_directory_list() );

    LIST *appaserver_upgrade_application_list(
        appaserver_upgrade_application_name_list(),
        appaserver_upgrade_version_list() );

    LIST *appaserver_upgrade_execute_list(
        first_boolean,
        appaserver_upgrade_application_name_list,
        appaserver_upgrade_directory_list(),
        appaserver_upgrade_application_list() );
}
```



APPASERVER_UPGRADE (2)

```
/* Usage */
char *appaserver_upgrade_root(
    char *mount_point );

/* Process */
sprintf(
    static root[],
    "%s/upgrade",
    mount_point );

/* Usage */
LIST *appaserver_upgrade_application_name_list(
    char *APPLICATION_LOG_EXTENSION,
    char *application_name,
    char *log_directory,
    LIST *appaserver_upgrade_application_list_argv() );

/* Process */
if ( application_name )
{
    LIST *application_name_list = list_new();

    list_set(
        application_name_list,
        application_name );
}
else
if ( list_length(
        appaserver_upgrade_application_list_argv ) )
{
    LIST *application_name_list =
        appaserver_upgrade_application_list_argv;
}
else
{
    LIST *application_name_list(
        APPLICATION_LOG_EXTENSION,
        log_directory );
}

/* Driver */
for APPASERVER_UPGRADE_EXECUTE *
    appaserver_upgrade_execute in
        appaserver_upgrade->
            appaserver_upgrade_execute_list )
{
    if ( !list_length(
            appaserver_upgrade_execute->
                system_string_list ) )
    {
        continue;
    }

    void environment_database_set(
        appaserver_upgrade_execute->
            application_name );

    void appaserver_upgrade_execute_list_system(
        appaserver_upgrade_execute->
            application_name,
        appaserver_upgrade_execute->
            system_string_list,
        appaserver_upgrade_execute->
            insert_statement_list );
}

/* Attributes */
LIST *application_list_argv;
LIST *application_name_list;
char *root;
LIST *appaserver_upgrade_directory_list;
LIST *version_list;
LIST *appaserver_upgrade_application_list;
LIST *appaserver_upgrade_execute_list;
```

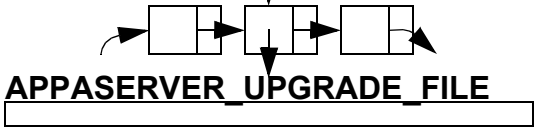


APPASERVER_UPGRADE_EXECUTE

```
char *application_name
LIST *system_string_list
LIST *insert_statement_list
```

APPASERVER_UPGRADE_DIRECTORY

```
/* Usage */
LIST *appaserver_upgrade_directory_list(
    char *appaserver_upgrade_root() );

/* Process */
LIST *list = list_new();

char *appaserver_upgrade_directory_system_string(
    appaserver_upgrade_root );

FILE *appaserver_input_pipe(
    appaserver_upgrade_directory_system_string() );

for char appaserver_version[] in
    string_input(
        appaserver_input_pipe() );
{
    APPASERVER_UPGRADE_DIRECTORY *
        appaserver_upgrade_directory =
            appaserver_upgrade_directory_fetch(
                appaserver_upgrade_root,
                strdup( appaserver_version ) );

    if ( appaserver_upgrade_directory )
    {
        list_set(
            list,
            appaserver_upgrade_directory );
    }
}
pclose( appaserver_input_pipe() );

/* Usage */
APPASERVER_UPGRADE_DIRECTORY *
    appaserver_upgrade_directory_fetch(
        char *appaserver_upgrade_root(),
        char *appaserver_version );

/* Process */
APPASERVER_UPGRADE_DIRECTORY *
    appaserver_upgrade_directory_calloc(
        void );

LIST *appaserver_upgrade_file_list(
    appaserver_upgrade_root,
    appaserver_version );

if ( !list_length( appaserver_upgrade_file_list() )
        return NULL;

/* Usage */
char *appaserver_upgrade_directory_system_string(
    char *appaserver_upgrade_root() );

/* Process */
sprintf(
    static system_string[],
    "cd %s && ls -1d * | sort_delimited_numbers.e '.' n",
    appaserver_upgrade_root );

/* Attributes */
char *appaserver_version;
LIST *appaserver_upgrade_file_list;
```



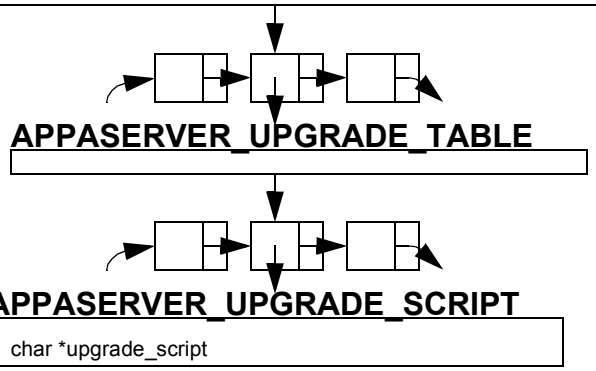
APPASERVER_UPGRADE_FILE

APPASERVER_UPGRADE_FILE

```
/* Usage */
LIST *appaserver_upgrade_file_list(
    char *appaserver_upgrade_root(),
    char *appaserver_version );

/* Process */
LIST *list = list_new();

char *appaserver_upgrade_file_root(
    appaserver_upgrade_root,
    appaserver_version );

char *appaserver_upgrade_file_system_string(
    appaserver_upgrade_file_root() );

FILE *appaserver_input_pipe(
    appaserver_upgrade_file_system_string() );

for char upgrade_script[] in
    string_input(
        appaserver_input_pipe() )
{
    APPASERVER_UPGRADE_FILE *
        appaserver_upgrade_file =
            appaserver_upgrade_file_new(
                appaserver_version,
                appaserver_upgrade_file_root(),
                strdup( upgrade_script ) );
    list_set(
        list,
        appaserver_upgrade_file );
}
pclose( appaserver_input_pipe() );

/* Usage */
char *appaserver_upgrade_file_root(
    char *appaserver_upgrade_root(),
    char *appaserver_version );

/* Process */
sprintf(
    static root[],
    "%s/%s",
    appaserver_upgrade_root,
    appaserver_version );

/* Usage */
char *appaserver_upgrade_file_system_string(
    char *appaserver_upgrade_file_root() );

/* Process */
sprintf(
    static system_string[],
    "cd %s && ls -1 *",
    appaserver_upgrade_file_root );

/* Usage */
APPASERVER_UPGRADE_FILE *
    appaserver_upgrade_file_new(
        char *appaserver_version,
        char *appaserver_upgrade_file_root(),
        char *upgrade_script );

/* Process */
APPASERVER_UPGRADE_FILE *
    appaserver_upgrade_file_calloc(
        void );

char *appaserver_upgrade_file_execute_system_string(
    char *appaserver_upgrade_file_root,
    char *upgrade_script );

char *appaserver_upgrade_file_insert_statement(
    char *APPASERVER_UPGRADE_SCRIPT_TABLE,
    char *APPASERVER_UPGRADE_SCRIPT_INSERT,
    char *appaserver_version,
    char *upgrade_script );


/* Attributes */
char *upgrade_script;
char *execute_system_string;
char *insert_statement;
```

APPASERVER_UPGRADE_APPLICATION

```
/* Usage */
LIST *appaserver_upgrade_application_list(
    LIST *appaserver_upgrade_application_name_list(),
    LIST *appaserver_upgrade_version_list() );

/* Process */
LIST *list = list_new();

for char *application_name in
    appaserver_upgrade_application_name_list
{
    APPASERVER_UPGRADE_APPLICATION *
        appaserver_upgrade_application =
            appaserver_upgrade_application_new(
                application_name,
                appaserver_upgrade_version_list );

    list_set(
        list,
        appaserver_upgrade_application );
}

/* Usage */
APPASERVER_UPGRADE_APPLICATION *
    appaserver_upgrade_application_new(
        char *application_name,
        LIST *appaserver_upgrade_version_list() );

/* Process */
APPASERVER_UPGRADE_APPLICATION *
    appaserver_upgrade_application_calloc(
        void );

LIST *appaserver_upgrade_table_list(
    application_name,
    appaserver_upgrade_version_list );

/* Attributes */
char *application_name;
LIST *appaserver_upgrade_table_list;
```

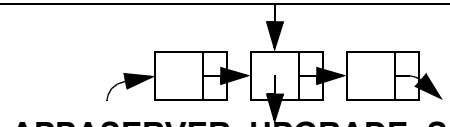


APPASERVER_UPGRADE_TABLE

```
/* Usage */
LIST *appaserver_upgrade_table_list(
    char *application_name,
    LIST *appaserver_upgrade_version_list );

/* Process */
LIST *list = list_new();

void environment_database_set(
    application_name );

for char *appaserver_version in
    appaserver_upgrade_version_list
{
```

```
    APPASERVER_UPGRADE_TABLE *
        appaserver_upgrade_table =
            appaserver_upgrade_table_new(
                appaserver_version );

    list_set(
        list,
        appaserver_upgrade_table );
}

/* Usage */
APPASERVER_UPGRADE_TABLE *
    appaserver_upgrade_table_new(
        char *appaserver_version );
```

```
/* Process */
APPASERVER_UPGRADE_TABLE *
    appaserver_upgrade_table_calloc(
        void );

LIST *appaserver_upgrade_script_list(
    appaserver_version );

/* Attributes */
char *appaserver_version;
LIST *appaserver_upgrade_script_list;
```



APPASERVER_UPGRADE_SCRIPT

APPASERVER_UPGRADE_SCRIPT

```
/* Usage */
LIST *appaserver_upgrade_script_list(
    char *appaserver_version );

/* Process */
LIST *list = list_new();

char *appaserver_upgrade_script_where(
    char *appaserver_version );

char *appaserver_system_string(
    "upgrade_script" /* select */,
    APPASERVER_UPGRADE_SCRIPT_TABLE,
    appaserver_upgrade_script_where() );

FILE *appaserver_input_pipe(
    appaserver_system_string() );

for char upgrade_script[] in
    string_input(
        appaserver_input_pipe() );
{
    APPASERVER_UPGRADE_SCRIPT *
        appaserver_upgrade_script =
            appaserver_upgrade_script_new(
                appaserver_version,
                strdup( upgrade_script ) );

    list_set(
        list,
        appaserver_upgrade_script );
}
pclose( appaserver_input_pipe() );


/* Usage */
APPASERVER_UPGRADE_SCRIPT *
    appaserver_upgrade_script_new(
        char *appaserver_version,
        char *upgrade_script );

/* Process */
APPASERVER_UPGRADE_SCRIPT *
    appaserver_upgrade_script_calloc(
        void );

/* Usage */
boolean appaserver_upgrade_script_exists(
    char *application_name,
    char *appaserver_version,
    char *upgrade_script,
    LIST *appaserver_upgrade_application_list() );

/* Process */
for APPASERVER_UPGRADE_APPLICATION *
    appaserver_upgrade_application in
        appaserver_upgrade_application_list
{
    if ( strcmp(
            appaserver_upgrade_application->
                application_name,
            application_name ) != 0 )
    {
        continue;
    }

    for APPASERVER_UPGRADE_TABLE *
        appaserver_upgrade_table in
            appaserver_upgrade_application->
                appaserver_upgrade_table_list
    {
        if ( strcmp(
                appaserver_upgrade_table->
                    appaserver_version,
                    appaserver_version ) != 0 )
        {
            continue;
        }

        APPASERVER_UPGRADE_SCRIPT *
            appaserver_upgrade_script_seek(
                char *upgrade_script,
                LIST *appaserver_upgrade_table->
                    appaserver_upgrade_script_list );

        if ( appaserver_upgrade_script_seek() )
            return 1;
        else
            return 0;
    }
}
return 0;

/* Attributes */
char *upgrade_script;
```

APPASERVER_UPGRADE_EXECUTE

```
/* Usage */
LIST *appaserver_upgrade_execute_list(
    boolean step_boolean,
    LIST *appaserver_upgrade_application_name_list,
    LIST *appaserver_upgrade_directory_list(),
    LIST *appaserver_upgrade_application_list() );

/* Process */
LIST *list = list_new();

for char *application_name in
    appaserver_upgrade_application_name_list
{
    APPASERVER_UPGRADE_EXECUTE *
        appaserver_upgrade_execute =
            appaserver_upgrade_execute_new(
                step_boolean,
                application_name,
                appaserver_upgrade_directory_list,
                appaserver_upgrade_application_list );

    if ( appaserver_upgrade_execute )
    {
        list_set(
            list,
            appaserver_upgrade_execute );
    }
}

/* Usage */
APPASERVER_UPGRADE_EXECUTE *
    appaserver_upgrade_execute_new(
        boolean step_boolean,
        char *application_name,
        LIST *appaserver_upgrade_directory_list,
        LIST *appaserver_upgrade_application_list );

/* Process */
APPASERVER_UPGRADE_EXECUTE *
    appaserver_upgrade_execute_calloc(
        void );

LIST *system_string_list = list_new();
LIST *insert_statement_list = list_new();
boolean all_done = 0;

for APPASERVER_UPGRADE_DIRECTORY *
    appaserver_upgrade_directory in
        appaserver_upgrade_directory_list
{
    for APPASERVER_UPGRADE_FILE *
        appaserver_upgrade_file in
            appaserver_upgrade_file_list )
    {
        boolean appaserver_upgrade_script_exists(
            application_name,
            appaserver_upgrade_directory->
                appaserver_version,
            appaserver_upgrade_file->upgrade_script,
            appaserver_upgrade_application_list );

        if ( appaserver_upgrade_script_exists() ) continue;

        list_set(
            system_string_list,
            appaserver_upgrade_file->
                execute_system_string );

        list_set(
            insert_statement_list,
            appaserver_upgrade_file->
                insert_statement );

        if ( step_boolean )
        {
            all_done = 1;
            break;
        }
    }
    if ( all_done ) break;
}

if ( !list_length( system_string_list ) ) return NULL;

/* Usage */
void appaserver_upgrade_execute_list_system(
    char *application_name,
    LIST *system_string_list,
    LIST *insert_statement_list );

/* Process */
for char *system_string in system_string_list
{
    printf(
        "%s: %s\n",
        application_name,
        system_string );
    fflush( stdout );

    system( system_string );
    fflush( stdout );
}

FILE *appaserver_output_pipe( "sql.e" );

for char *insert_statement in insert_statement_list
{
    fprintf(    appaserver_output_pipe(),
                "%s\n",
                insert_statement );
}
pclose( appaserver_output_pipe() );

/* Attributes */
char *application_name;
LIST *system_string_list;
LIST *insert_statement_list;
```

GRANT_SELECT (1)

```
/* Usage */
GRANT_SELECT *grant_select_new(
    char *application_name,
    char *security_sql_injection_escape(
        login_name ),
    char *security_sql_injection_escape(
        connect_from_host ),
    boolean revoke_only_boolean );

/* Process */
GRANT_SELECT *grant_select_calloc(
    void );

char *grant_select_login_host_name(
    login_name,
    connect_from_host );

char *grant_select_flush_privileges_sql_statement(
    void );

if ( revoke_only_boolean )
{
    char *grant_select_revoke_sql_statement(
        grant_select_login_host_name() );

    char *grant_select_drop_user_sql_statement(
        grant_select_login_host_name() );

    return this;
}

char *security_generate_password();

char *grant_select_create_user_sql_statement(
    grant_select_login_host_name(),
    security_generate_password() );

LIST *grant_select_on_table_sql_statement_list(
    application_name,
    grant_select_login_host_name() );

char *grant_select_update_process_sql_statement(
    application_name,
    grant_select_login_host_name() );

char *grant_select_post_execute_message(
    char *application_name,
    char *connect_from_host,
    char *security_generate_password() );

/* Driver */
void grant_select_display(
    char *create_user_sql_statement,
    LIST *on_table_sql_statement_list,
    char *update_process_sql_statement,
    char *flush_privileges_sql_statement )

/* Process */

/* Driver */
void grant_select_execute(
    char *create_user_sql_statement,
    LIST *on_table_sql_statement_list,
    char *update_process_sql_statement,
    char *flush_privileges_sql_statement );

/* Process */

/* Driver */
void grant_select_revoke_display(
    char *revoke_sql_statement,
    char *drop_user_sql_statement,
    char *flush_privileges_sql_statement );

/* Process */

/* Driver */
void grant_select_revoke_execute(
    char *revoke_sql_statement,
    char *drop_user_sql_statement,
    char *flush_privileges_sql_statement );

/* Process */

/* Attributes */
char *login_host_name;
char *flush_privileges_sql_statement;
char *revoke_sql_statement;
char *drop_user_sql_statement;
char *security_generate_password;
char *create_user_sql_statement;
LIST *on_table_sql_statement_list;
char *update_process_sql_statement;
char *post_execute_message;
```

GRANT_SELECT (2)

```
/* Usage */
char *grant_select_login_host_name(
    char *login_name,
    char *connect_from_host );

/* Process */
sprintf(
    static login_host_name[],
    "'%s'@'%s'",
    login_name,
    connect_from_host );

/* Usage */
char *grant_select_drop_user_sql_statement(
    char *grant_select_login_host_name );

/* Process */
sprintf( static sql_statement[],
         "drop user if exists %s;",
         grant_select_login_host_name );

/* Usage */
char *grant_select_create_user_sql_statement(
    char *grant_select_login_host_name,
    char *security_generate_password );

/* Process */
sprintf( static sql_statement[],
         "create user %s identified by '%s';",
         grant_select_login_host_name,
         security_generate_password );

/* Usage */
LIST *grant_select_on_table_sql_statement_list(
    char *application_name,
    char *grant_select_login_host_name );

/* Process */
LIST *list = list_new();

LIST *application_non_system_folder_name_list();

for char *folder_name in
    application_non_system_folder_name_list()
{
    list_set(
        list,
        char *grant_select_on_table_sql_statement(
            application_name,
            grant_select_login_host_name,
            folder_name ) );
}

/* Usage */
char *grant_select_on_table_sql_statement(
    char *application_name,
    char *grant_select_login_host_name,
    char *folder_name );

/* Process */
sprintf( sql_statement[],
         "grant select on %s.%s to %s;",
         application_name,
         folder_name,
         grant_select_login_host_name );

/* Usage */
char *grant_select_update_process_sql_statement(
    char *application_name,
    char *grant_select_login_host_name );

/* Process */
sprintf( static sql_statement[],
         "grant update (execution_count) "
         "on %s.process "
         "to %s;",
         application_name,
         grant_select_login_host_name );

/* Usage */
char *grant_select_revoke_sql_statement(
    char *grant_select_login_host_name );

/* Process */
sprintf( static sql_statement[],
         "revoke all on *.* from %s;",
         grant_select_login_host_name );
```

GOOGLE_MAP (1)

```
/* Usage */
char *google_map_key(
    char *GOOGLE_MAP_KEY_LABEL );

/* Process */
return
application_constant_fetch(
    GOOGLE_MAP_KEY_LABEL );

/* Usage */
char *google_map_initialize_javascript(
    char *GOOGLE_MAP_CANVAS_LABEL,
    char *GOOGLE_MAP_CENTER_LATITUDE,
    char *GOOGLE_MAP_CENTER_LONGITUDE,
    int GOOGLE_MAP_STARTING_ZOOM );

/* Process */
sprintf( script[],
"function initialize()\n"
"{\n"
"   var point;\n"
"   var the_center = "
"new google.maps.LatLng(%s,%s);\n"
"\n"
"   var mapOptions = {\n"
"       center: the_center,\n"
"       zoom: %d,\n"
"       mapTypeId: google.maps.MapTypeId.roadmap\n"
"};\n"
"\n"
"   var map = new google.maps.Map(\n"
"       document.getElementById(\"%s\"),\n"
"       mapOptions);\n"
"\n",
GOOGLE_MAP_CENTER_LATITUDE,
GOOGLE_MAP_CENTER_LONGITUDE,
GOOGLE_MAP_STARTING_ZOOM,
GOOGLE_MAP_CANVAS_LABEL );

/* Usage */
char *google_map_script_open_tag(
    void );

/* Process */
return
"<script type=\"text/javascript\">\n"
"\n"
"//<![CDATA[\n"
"\n";

/* Usage */
char *google_map_script_close_tag(
    void );

/* Process */
return
"}\n"
"//]]>\n"
"</script>\n";

/* Usage */
char *google_map_body_html(
    char *GOOGLE_MAP_CANVAS_LABEL );

/* Process */
sprintf(
    html[],
    "<body onload=\"initialize()\" "
    "leftmargin=0 "
    "topmargin=0 "
    "marginwidth=0 "
    "marginheight=0>\n"
    "<div id=\"%s\"></div>\n"
    "</body>\n",
    GOOGLE_MAP_CANVAS_LABEL );
```

GOOGLE_MAP (2)

```
/* Usage */
char *google_map_canvas_style_html(
    char *GOOGLE_MAP_CANVAS_LABEL,
    int absolute_position_top,
    int absolute_position_left,
    int google_map_width,
    int google_map_height );

/* Process */
char style_html[];
char *ptr = style_html;

ptr += sprintf( ptr,
“<style type=\”text/css\”>\n”
“#%s {\n”
“   width: %dpx;\n”
“   height: %dpx;\n”,
    GOOGLE_MAP_CANVAS_LABEL,
    google_map_width,
    google_map_height );

if ( absolute_position_top + absolute_position_left )
{
    ptr += sprintf( ptr,
“   position: absolute;\n”
“   top: %dpx;\n”
“   left: %dpx;\n”,
    absolute_position_top,
    absolute_position_left );
}

sprintf( ptr,
“}\n”
“</style>\n” );

/* Usage */
char *google_map_loader_script_html(
    char *google_map_key() );

/* Process */
snprintf(
    html[],
    sizeof ( html ),
"<script async src=\"https://maps.googleapis.com/
maps/api/js?key=%s&callback=console.debug&librar-
ies=maps,marker&v=beta\"></script>",
    google_map_key );

/* Usage */
char *google_map_create_marker_javascript(
    void );

/* Process */
return
“var infowindow;\n”
"function create_selectable_map_marker( map, point,
click_message )\n"
"{\n"
"   var content;\n"
"   var marker;\n"
"\n"
"   content = '<u style=\"font-style: 2em;\"
onclick=onclick_selectable_function(\"' +
click_message + '\")>' + click_message + '</u>';\n"
"\n"
"   marker = new google.maps.Marker({\n"
"       position: point,\n"
"       map: map\n"
"});\n"
"\n"
"   infowindow = new google.maps.InfoWindow({\n"
"   content: '<br>' + click_message\n"
"});\n"
"\n"
"   google.maps.event.addListener(marker, 'click', func-
tion() {\n"
"   infowindow.setContent( content );\n"
"   infowindow.open( map, marker );\n"
"   });\n"
"}\n";

/* Usage */
char *google_map_click_message(
    char *message );

/* Process */
char *string_initial_capital(
    destination[],
    message );
```

GOOGLE_MAP_OPERATION (1)

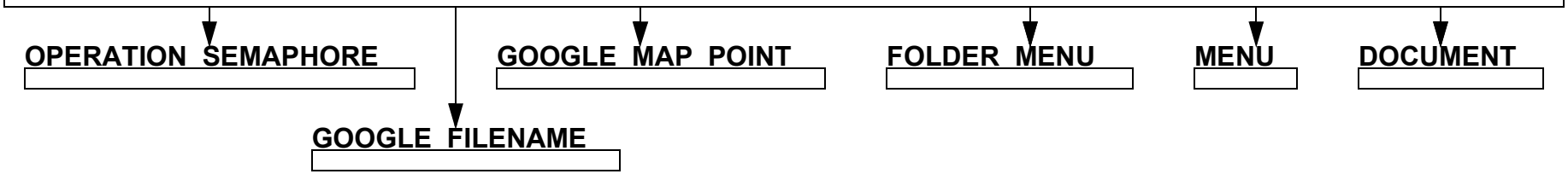
```
/* Usage */
GOOGLE_MAP_OPERATION *
   google_map_operation_new(
      char *application_name,
      char *session_key,
      char *login_name,
      char *role_name,
      char *folder_name,
      char *target_frame,
      char *operation_name,
      char *latitude_string,
      char *longitude_string,
      char *utm_easting_string,
      char *utm_northing_string,
      char *primary_data_list_string,
      pid_t parent_process_id,
      int operation_row_checked_count,
      char *appaserver_parameter->data_directory );

/* Process */
if ( !operation_row_checked_count ) return null;

GOOGLE_MAP_OPERATION *
   google_map_operation_calloc(
      void );

OPERATION_SEMAPHORE *
   operation_semaphore =
      operation_semaphore_new(
         application_name,
         operation_name,
         data_directory,
         parent_process_id,
         operation_row_checked_count );

GOOGLE_FILENAME *google_filename =
   google_filename_new(
      application_name,
      session_key,
      data_directory,
      (char *)0 /* number_attribute_name */,
      (char *)0 /* chart_filename_key */ );

FILE *google_map_operation_output_file(
   boolean operation_semaphore->group_first_time,
   char *google_filename->http_output_filename );

if ( operation_semaphore->group_first_time )
{
   char *google_map_key(
      GOOGLE_MAP_KEY_LABEL );

   char *google_map_operation_title_string(
      char *operation_name );

   char *google_map_canvas_style_html(
      GOOGLE_MAP_CANVAS_LABEL,
      0 /* absolute_position_top */,
      0 /* absolute_position_left */,
      GOOGLE_MAP_WIDTH,
      GOOGLE_MAP_HEIGHT );

   char *google_map_loader_script_html(
      google_map_key() );

   char *google_map_script_open_tag();

   char *google_map_create_marker_javascript();

   char *google_map_initialize_javascript(
      GOOGLE_MAP_CANVAS_LABEL,
      GOOGLE_MAP_CENTER_LATITUDE,
      GOOGLE_MAP_CENTER_LONGITUDE,
      GOOGLE_MAP_STARTING_ZOOM );

   char *google_map_operation_document_open_html(
      application_name,
      google_map_operation_title_string(),
      google_map_canvas_style_html(),
      google_map_loader_script_html(),
      google_map_script_open_tag(),
      google_map_create_marker_javascript(),
      google_map_initialize_javascript() );

   fprintf(
      google_map_output_file(),
      "%s\n",
      google_map_operation_document_open_html() );
}

char *google_map_click_message(
   primary_data_list_string /* message */ );

if ( string_atoi( latitude_string ) )
{
   GOOGLE_MAP_POINT *google_map_point =
      google_map_point_latitude_longitude_new(
         latitude_string,
         longitude_string,
         google_map_click_message() );
}
else
{
   GOOGLE_MAP_POINT *google_map_point =
      google_map_point_utm_new(
         utm_easting_string,
         utm_northing_string,
         google_map_click_message() );
}

fprintf(
   google_map_operation_output_file(),
   "\n%s\n",
   google_map_point->javascript );

if ( operation_semaphore->group_last_time )
{
   fprintf(
      google_map_operation_output_file(),
      "%s\n",
      google_map_script_close_tag() );

   fprintf(
      google_map_operation_output_file(),
      "%s\n",
      document_head_close_tag() );

   char *google_map_body_html(
      GOOGLE_MAP_CANVAS_LABEL );

   fprintf(
      google_map_operation_output_file(),
      "%s\n",
      google_map_body_html() );

   fprintf(
      google_map_operation_output_file(),
      "%s\n",
      document_close_tag() );
}

void fclose( google_map_operation_output_file() );
```

GOOGLE_MAP_OPERATION (2)

```
/* Process (continued) */
if ( operation_semaphore->group_last_time )
{
    boolean application_menu_horizontal_boolean(
        application_name );

    boolean menu_horizontal_boolean(
        target_frame /* current_frame */,
        FRAMESET_PROMPT_FRAME,
        application_menu_horizontal_boolean() );

    if ( menu_horizontal_boolean() )
    {
        FOLDER_MENU *folder_menu =
            folder_menu_fetch(
                application_name,
                session_key,
                role_name,
                data_directory,
                0 /* not folder_menu_remove_boolean */ );

        MENU *menu =
            menu_new(
                application_name,
                session_key,
                login_name,
                role_name,
                1 /* application_menu_horizontal_boolean */,
                folder_menu->count_list );
    }

    char *document_head_menu_setup_string(
        (menu) ? 1 : 0 /* menu_boolean */ );

    DOCUMENT *document =
        document_new(
            application_name,
            application_title_string( application_name ),
            google_map_operation_title_string(),
            (char *)0 /* sub_title_string*/,
            (char *)0 /* sub_sub_title_string */,
            (char *)0 /* notepad */,
            (char *)0 /* onload_javascript_string */,
            (menu) ? menu->html : (char *)0,
            document_head_menu_setup_string(),
            (char *)0 /* calendar_setup_string */,
            javascript_include_string(
                application_name,
                (LIST *)0 /* javascript_filename_list */ ) );

        char *document_form_html(
            document->html,
            document->document_head->html,
            document_head_close_tag(),
            document->document_body->html,
            (char *)0 /* form_html */,
            (char *)0 /* document_body_close_tag */,
            (char *)0 /* document_close_tag */ );
    }
```

GOOGLE_MAP_OPERATION (3)

```
/* Usage */
char *google_map_operation_document_open_html(
    char *application_name,
    char *google_map_operation_title_string(),
    char *google_map_canvas_style_html(),
    char *google_map_loader_script_html(),
    char *google_map_script_open_tag(),
    char *google_map_create_marker_javascript(),
    char *google_map_initialize_javascript() );

/* Process */
char html[];
char *ptr = html;

char *document_type_html();

char *document_standard_string();

ptr +=
    sprintf(
        ptr,
        "%s\n",
        document_html(
            document_type_html(),
            document_standard_string() ) );

DOCUMENT_HEAD *document_head =
    document_head_new(
        application_name,
        google_map_operation_title_string,
        (char *)0 /* menu_setup_string */,
        (char *)0 /* calendar_setup_string */,
        javascript_include_string(
            application_name,
            (LIST *)0 /* javascript_filename_list */ ) );

ptr +=
    sprintf(
        ptr,
        "%s\n",
        document_head->html );

ptr +=
    sprintf(
        ptr,
        "%s\n",
        google_map_canvas_style_html );

ptr +=
    sprintf(
        ptr,
        "%s\n",
        google_map_loader_script_html );

ptr +=
    sprintf(
        ptr,
        "%s\n",
        google_map_script_open_tag );

ptr +=
    sprintf(
        ptr,
        "%s\n",
        google_map_create_marker_javascript );

ptr +=
    sprintf(
        ptr,
        "%s",
        google_map_initialize_javascript );

/* Driver */
if ( google_map_operation->
        operation_semaphore->
        group_last_time )
{
    printf(
        "%s\n",
        google_map_operation->
            document->
            document_form_html );

    printf(
        "%s\n",
        google_map_operation->
            google_filename->
            appaserver_link_anchor_html );

    void document_close();
}

/* Attributes */
OPERATION_SEMAPHORE *operation_semaphore;
GOOGLE_FILENAME *google_filename;
FILE *output_file;
char *google_map_key;
char *title_string;
char *google_map_canvas_style_html;
char *google_map_loader_script_html;
char *google_map_script_open_tag;
char *google_map_create_marker_javascript;
char *google_map_initialize_javascript;
char *document_open_html;
char *google_map_click_message;
GOOGLE_MAP_POINT *google_map_point;
char *google_map_body_html;
boolean application_menu_horizontal_boolean;
boolean menu_horizontal_boolean;
FOLDER_MENU *folder_menu;
MENU *menu;
char *title_html;
char *document_head_menu_setup_string;
DOCUMENT *document;
char *document_form_html;
```



GOOGLE_MAP_POINT (1)

```
/* Usage */
GOOGLE_MAP_POINT *
   google_map_point_latitude_longitude_new(
      char *latitude_string,
      char *longitude_string,
      char *google_map_click_message() );

/* Process */
GOOGLE_MAP_POINT *google_map_point_calloc(
   void );

double latitude_nad83 =
   google_map_point_nad83(
      latitude_string /* latitude_longitude */ );

double longitude_nad83 =
   google_map_point_nad83(
      longitude_string /* latitude_longitude */ );

char *google_map_point_javascript(
   latitude_nad83,
   longitude_nad83,
   google_map_click_message );

/* Usage */
GOOGLE_MAP_POINT *
   google_map_point_utm_new(
      char *utm_easting_string,
      char *utm_northing_string,
      char *google_map_click_message() );

/* Process */
GOOGLE_MAP_POINT *google_map_point_calloc();

void google_map_point_cs2cs(
   &latitude_nad83,
   &longitude_nad83,
   utm_easting_string,
   utm_northing_string );

char *google_map_point_javascript(
   latitude_nad83,
   longitude_nad83,
   google_map_click_message );

/* Usage */
double google_map_point_nad83(
   char *latitude_longitude );

/* Process */
int string_character_count( ' ', latitude_longitude );

if ( !string_character_count() )
{
   double atof( latitude_longitude );
}
else
if ( string_character_count() == 1 )
{
   double google_map_point_degrees_minutes(
      latitude_longitude );
}
else
{
   double
      google_map_point_degrees_minutes_seconds(
         latitude_longitude );
}

char *google_map_point_javascript(
   latitude_nad83,
   longitude_nad83,
   google_map_click_message );

/* Usage */
double google_map_point_degrees_minutes(
   char *latitude_longitude );

/* Process */

/* Usage */
double google_map_point_degrees_minutes_seconds(
   char *latitude_longitude );

/* Process */
```

GOOGLE_MAP_POINT (2)

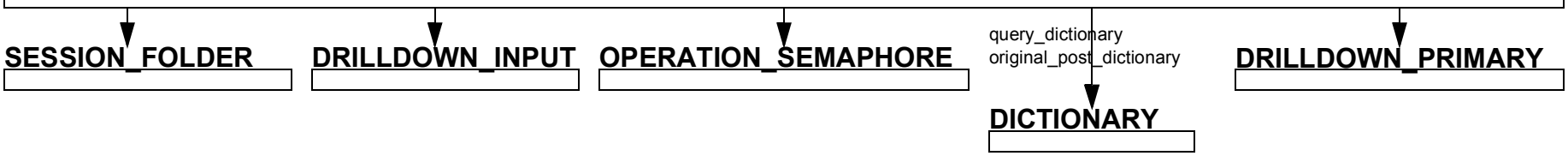
```
/* Usage */
void google_map_point_cs2cs(
    double *latitude_nad83,
    double *longitude_nad83,
    char *utm_easting_string,
    char *utm_northing_string );

/* Process */
char *google_map_point_cs2cs_system_string(
    utm_easting_string,
    utm_northing_string );

char *string_pipe_fetch(
    google_map_point_cs2cs_system_string() );

char *column(
    longitude_latitude_string[],
    0,
    string_pipe_fetch() );

*longitude_nad83 = atof( column() );

char *column(
    longitude_latitude_string[],
    1,
    string_pipe_fetch() );

*latitude_nad83 = atof( column() );

/* Usage */
char *google_map_point_cs2cs_system_string(
    char *utm_easting_string,
    char *utm_northing_string );

/* Process */
sprintf(
    static system_string[],
    “echo \”%s %s\” | cs2cs_utm.sh”,
    utm_easting_string,
    utm_northing_string );

/* Usage */
char *google_map_point_javascript(
    double latitude_nad83,
    double longitude_nad83,
    char *google_map_click_message );

/* Process */
sprintf(javascript[],
"   point = new google.maps.LatLng(%.7lf,%.7lf);\n",
"   create_selectable_map_marker( map, point, '%s'
);\n\n",
    latitude_nad83,
    longitude_nad83 );
    google_map_click_message );

/* Attributes */
double latitude_nad83;
double longitude_nad83;
char *javascript;
```

UPLOAD_SOURCE

```
/* Usage */
UPLOAD_SOURCE *upload_source_new(
    char *application_name,
    char *directory_filename_session,
    char *appaserver_parameter->mount_point );

/* Process */
boolean upload_source_file_exists_boolean(
    directory_filename_session );

if ( !upload_source_file_exists_boolean() ) return this;

char *upload_source_directory_filename(
    directory_filename_session );

char *upload_source_filename(
    upload_source_directory_filename() );

char *upload_source_destination_directory(
    application_name,
    mount_point );

boolean destination_directory_exists_boolean =
    upload_source_file_exists_boolean(
        upload_source_destination_directory()
            /* directory_filename_session */ );

if ( !destination_directory_exists_boolean ) return this;

char *upload_source_destination_filename(
    upload_source_filename(),
    upload_source_destination_directory() );

char *upload_source_copy_system_string(
    directory_filename_session,
    upload_source_destination_filename() );

char *upload_source_chmod_system_string(
    upload_source_destination_filename() );

char *upload_source_confirm_system_string(
    upload_source_filename(),
    upload_source_destination_directory() );

/* Usage */
boolean upload_source_file_exists_boolean(
    char *directory_filename_session );

/* Process */
sprintf( system_string,
    "echo %s | stat.e 2>&1 | grep 'No such file' | wc -l",
    directory_filename_session );

return 1 - atoi( string_pipe_fetch( system_string ) );

/* Usage */
char *upload_source_directory_filename(
    char *directory_filename_session );

/* Process */
strcpy(
    static directory_filename[],
    directory_filename_session );

char *session_right_trim(
    directory_filename
        /* directory_filename_session_suffix  */ );

/* Usage */
char *upload_source_filename(
    char *upload_source_directory_filename() );

/* Process */
char *basename_filename(
    upload_source_directory_filename /* argv_0 */ );

/* Usage */
char *upload_source_destination_directory(
    char *application_name,
    char *mount_point );

/* Process */
sprintf( static directory[],
    "%s/src_%s",
    mount_point,
    application_name );

/* Usage */
char *upload_source_destination_filename(
    char *upload_source_filename(),
    char *upload_source_destination_directory() );

/* Process */
sprintf( static filename[],
    "%s/%s",
    upload_source_destination_directory,
    upload_source_filename );

/* Usage */
char *upload_source_copy_system_string(
    char *directory_filename_session,
    char *upload_source_destination_filename() );

/* Process */
sprintf( static system_string[],
    "cp %s %s",
    directory_filename_session,
    upload_source_destination_filename );

/* Usage */
char *upload_source_chmod_system_string(
    char *upload_source_destination_filename() );

/* Process */
sprintf( static system_string[],
    "chmod g+w %s && chmod -x %s",
    upload_source_destination_filename,
    upload_source_destination_filename );

/* Usage */
char *upload_source_confirm_system_string(
    char *upload_source_filename(),
    char *upload_source_destination_directory() );

/* Process */
sprintf( static system_string[],
    "(cd %s && ls -l %s) 2>&1 | "
    "html_paragraph_wrapper.e",
    upload_source_destination_directory,
    upload_source_filename );

/* Driver */
system( upload_source->copy_system_string );
system( upload_source->chmod_system_string );
system( upload_source->confirm_system_string );

/* Attributes */
boolean file_exists_boolean;
char *directory_filename;
char *filename;
char *destination_directory;
boolean destination_directory_exists_boolean;
char *destination_filename;
char *copy_system_string;
char *chmod_system_string;
char *confirm_system_string;
```

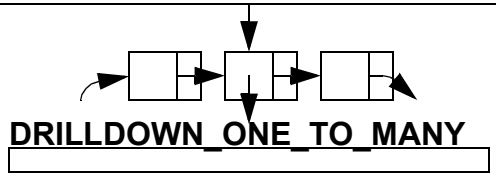
DRILLDOWN (1)

```
/* Usage */
DRILLDOWN *drilldown_new(
    int argc,
    char **argv,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *drilldown_base_folder_name,
    char *target_frame,
    char *drilldown_primary_data_list_string,
    char *update_results_string,
    char *update_error_string,
    pid_t parent_process_id,
    int operation_row_checked_count,
    DICTIONARY *post_dictionary->
        original_post_dictionary,
    char *appaserver_parameter->data_directory );

/* Process */
DRILLDOWN *drilldown_calloc(
    void );

SESSION_FOLDER *session_folder =
      /* Sets appaserver environment and outputs argv */
      /* Each parameter is security inspected */
      session_folder_integrity_exit(
          argc,
          argv,
          application_name,
          session_key,
          login_name,
          role_name,
          drilldown_base_folder_name,
          APPASERVER_LOOKUP_STATE,
          APPASERVER_UPDATE_STATE );

DRILLDOWN_INPUT *drilldown_input =
    drilldown_input_new(
        application_name,
        session_key,
        login_name,
        role_name,
        drilldown_base_folder_name,
        target_frame,
        drilldown_primary_data_list_string,
        data_directory,
        original_post_dictionary );

OPERATION_SEMAPHORE *
    operation_semaphore =
        operation_semaphore_new(
            application_name,
            DRILLDOWN_OPERATION_NAME,
            data_directory,
            parent_process_id,
            operation_row_checked_count );

DICTIONARY *drilldown_query_dictionary(
    QUERY_RELATION_OPERATOR_PREFIX,
    QUERY_EQUAL,
    drilldown_input->
        dictionary_separate->
        query_dictionary,
    drilldown_input->
        folder->
        folder_attribute_primary_key_list,
    drilldown_input->
        primary_attribute_data_list );

DICTIONARY *drilldown_original_post_dictionary(
    DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
    DICTIONARY_SEPARATE_SORT_PREFIX,
    DICTIONARY_SEPARATE_QUERY_PREFIX,
    drilldown_input->
        dictionary_separate->
        no_display_dictionary,
    drilldown_input->
        dictionary_separate->
        sort_dictionary,
    drilldown_query_dictionary() );

DRILLDOWN_PRIMARY *drilldown_primary_new(
    application_name,
    session_key,
    login_name,
    role_name,
    drilldown_base_folder_name,
    target_frame,
    drilldown_primary_data_list_string,
    data_directory,
    drilldown_input->process_id,
    drilldown_original_post_dictionary() );
```



DRILLDOWN (2)

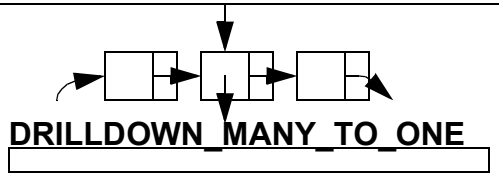
```
/* Process (continued) */
if ( list_rewind( drilldown_input->relation_one2m_list ) )
{
    LIST *drilldown_one_to_many_list = list_new();

    for RELATION_ONE2M *relation_one2m in
        drilldown_input->relation_one2m_list )
    {
        if ( relation_one2m->relation->omit_drilldown )
            continue;

        DRILLDOWN_ONE_TO_MANY *
            drilldown_one_to_many_new(
                application_name,
                session_key,
                login_name,
                role_name,
                drilldown_base_folder_name,
                target_frame,
                drilldown_primary_data_list_string,
                data_directory,
                drilldown_input->process_id,
                drilldown_input->
                    dictionary_separate->
                    no_display_dictionary,
                drilldown_input->
                    dictionary_separate->
                    sort_dictionary,
                drilldown_input->primary_attribute_data_list,
                relation_one2m->many_folder_name,
                relation_one2m->
                    relation_foreign_key_list );

        if ( drilldown_one_to_many_new() )
        {
            list_set(
                drilldown_one_to_many_list,
                drilldown_one_to_many_new() );
        }
    }
}
```



DRILLDOWN_ONE_TO_MANY

DRILLDOWN (3)

```
/* Process (continued) */
if ( list_length(
        drilldown_input->
           relation_mto1_isa_list ) )
{
   RELATION_MTO1 *relation_mto1 =
      list_first(
         drilldown_input->
            relation_mto1_isa_list );

   LIST *drilldown_many_to_one_list = list_new();

   DRILLDOWN_MANY_TO_ONE *
      drilldown_many_to_one_new(
         application_name,
         session_key,
         login_name,
         role_name,
         drilldown_base_folder_name,
         target_frame,
         drilldown_primary_data_list_string,
         data_directory,
         drilldown_input->process_id,
         drilldown_input->
            dictionary_separate->
            sort_dictionary,
         relation_mto1->one_folder_name,
         relation_mto1->relation_foreign_key_list,
         relation_mto1->relation_translate_list,
         relation_mto1->
            one_folder->
            folder_attribute_primary_key_list,
         drilldown_query_dictionary() );

   if ( drilldown_many_to_one_new() )
   {
      list_set(
         drilldown_many_to_one_list,
         drilldown_many_to_one_new() );

      drilldown_input->relation_mto1_list =
         relation_mto1_to_one_fetch_list(
            relation_mto1->one_folder_name,
            relation_mto1->
               one_folder->
               folder_attribute_primary_key_list );
   }
}

if ( list_rewind(
        drilldown_input->
           relation_mto1_to_list ) )
{
   char *appaserver_table_name(
      drilldown_base_folder_name );

   /* Fetch the foreign key; it may not be primary */
   DICTIONARY *query_fetch_dictionary(
      drilldown_input->
         folder->
         folder_attribute_name_list
            /* select_name_list */,
      appaserver_table_name(),
      drilldown_input->
         folder->
         folder_attribute_primary_key_list,
      drilldown_input->
         primary_attribute_data_list );

   /* A primary attribute was changed */
   if ( !query_fetch_dictionary() )
   {
      goto drilldown_document;
   }

   if ( !drilldown_many_to_one_list )
   {
      LIST *drilldown_many_to_one_list = list_new();
   }

   for RELATION_MTO1 *relation_mto1 in
      drilldown_input->relation_mto1_list )
   {
      DRILLDOWN_MANY_TO_ONE *
         drilldown_many_to_one_new(
            application_name,
            session_key,
            login_name,
            role_name,
            drilldown_base_folder_name,
            target_frame,
            drilldown_primary_data_list_string,
            data_directory,
            drilldown_input->process_id,
            drilldown_input->
               dictionary_separate->
               sort_dictionary,
            relation_mto1->one_folder_name,
            relation_mto1->relation_foreign_key_list,
            relation_mto1->relation_translate_list,
            relation_mto1->
               one_folder->
               folder_attribute_primary_key_list,
            query_fetch_dictionary() );

      if ( drilldown_many_to_one_new() )
      {
         list_set(
            drilldown_many_to_one_list,
            drilldown_many_to_one_new() );
      }
   }
}
```



DRILLDOWN (4)

```
/* Process (continued) */
drilldown_document:

DRILLDOWN_DOCUMENT *drilldown_document =
    drilldown_document_new(
        application_name,
        session_key,
        login_name,
        role_name,
        drilldown_base_folder_name,
        update_results_string,
        update_error_string,
        drilldown_input->menu,
        drilldown_input->
            folder->
            notepad,
        drilldown_input->
            folder->
            javascript_filename,
        drilldown_input->
            folder->
            post_change_javascript,
        drilldown_input->relation_mto1_list,
        drilldown_input->relation_one2m_list,
        drilldown_primary,
        drilldown_one_to_many_list );
```

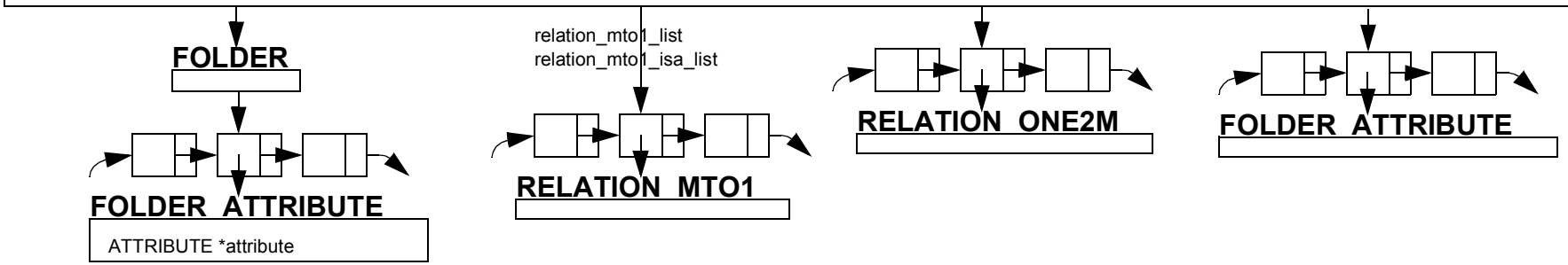
```
/* Usage */
DICTIONARY *drilldown_query_dictionary(
    const char *QUERY_RELATION_OPERATOR_PREFIX,
    const char *QUERY_EQUAL,
    DICTIONARY *dictionary_separate_query_dictionary,
    LIST *key_list,
    LIST *attribute_data_list );

/* Process */
DICTIONARY *attribute_query_dictionary(
    QUERY_RELATION_OPERATOR_PREFIX,
    QUERY_EQUAL,
    key_list,
    attribute_data_list );

if ( !dictionary_separate_query_dictionary )
{
    dictionary_separate_query_dictionary =
        dictionary_small();
}

void dictionary_set_dictionary(
    dictionary_separate_query_dictionary /* in/out */,
    attribute_query_dictionary() );

/* Usage */
void drilldown_table_edit_output(
    const char *APPASERVER_USER_LOGIN_NAME,
    char *login_name,
    TABLE_EDIT *table_edit );

/* Process */
void table_edit_spool_file_output(
    table_edit->appaserver_update_filespecification,
    table_edit->
        query_table_edit->
        query_fetch->
        row_list /* query_fetch_row_list */ );

void table_edit_output(
    APPASERVER_USER_LOGIN_NAME,
    login_name,
    table_edit->
        table_edit_input->
        folder_row_level_restriction->
        non_owner_viewonly,
    (char *)0 /* document_html */,
    table_edit->form_table_edit->open_html,
    table_edit->
        form_table_edit->
        heading_container_string,
    table_edit->
        table_edit_input->
        folder_operation_list,
    table_edit->
        query_table_edit->
        query_fetch->
        row_list /* query_fetch_row_list */,
    table_edit->relation_join,
    table_edit->
        row_security->
        regular_widget_list,
    table_edit->
        row_security->
        viewonly_widget_list,
    table_edit->
        table_edit_input->
        row_security_role_update_list,
    table_edit->
        table_edit_input->
        table_edit_state,
    table_edit->form_table_edit->close_html,
    (char *)0 /* document_trailer_html */ );

/* Usage */
DICTIONARY *drilldown_original_post_dictionary(
    const char *
        DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
    const char *
        DICTIONARY_SEPARATE_SORT_PREFIX,
    const char *
        DICTIONARY_SEPARATE_QUERY_PREFIX,
    DICTIONARY *no_display_dictionary,
    DICTIONARY *sort_dictionary,
    DICTIONARY *drilldown_query_dictionary );

/* Process */
DICTIONARY *original_post_dictionary =
    dictionary_medium();

void dictionary_prefixed_dictionary_set(
    original_post_dictionary /* out */,
    DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
    no_display_dictionary );

void dictionary_prefixed_dictionary_set(
    original_post_dictionary /* out */,
    DICTIONARY_SEPARATE_SORT_PREFIX,
    sort_dictionary );

void dictionary_prefixed_dictionary_set(
    original_post_dictionary /* out */,
    DICTIONARY_SEPARATE_QUERY_PREFIX,
    drilldown_query_dictionary );
```

DRILLDOWN (6)

```
/* Driver */
if ( drilldown->operation_semaphore->group_first_time )
{
    printf(
        "%s\n%s\n%s\n%s\n%s\n",
        drilldown->
            drilldown_document->
            document->
            html,
        drilldown->
            drilldown_document->
            document->
            document_head->
            html,
        drilldown->
            drilldown_document->
            ajax_client_list_javascript,
        document_head_close_tag(),
        drilldown->
            drilldown_document->
            document->
            document_body->
            html );
}

drilldown_table_edit_output(
    APPASERVER_USER_LOGIN_NAME,
    login_name,
    drilldown->drilldown_primary->table_edit );

printf( "<br><hr>\n" );

for DRILLDOWN_ONE_TO_MANY *
    drilldown_one_to_many in
        drilldown->drilldown_one_to_many_list
{
    printf( "<h2>%s</h2>\n",
        string_initial_capital(
            buffer[],
            drilldown_one_to_many->
                relation_many_folder_name );

    void drilldown_table_edit_output(
        APPASERVER_USER_LOGIN_NAME,
        login_name,
        drilldown_one_to_many->table_edit );

    printf( "<br><hr>\n" );
}

for DRILLDOWN_MANY_TO_ONE *
    drilldown_many_to_one in
        drilldown->drilldown_many_to_one_list
{
    printf( "<h2>%s</h2>\n",
        string_initial_capital(
            buffer[],
            drilldown_many_to_one->
                one_folder_name );

    void drilldown_table_edit_output(
        APPASERVER_USER_LOGIN_NAME,
        login_name,
        drilldown_many_to_one->table_edit );

    printf( "<br><hr>\n" );
}

if ( drilldown->operation_semaphore->group_last_time )
{
    void document_close();
}

/* Attributes */
SESSION_FOLDER *session_folder;
DRILLDOWN_INPUT *drilldown_input;
OPERATION_SEMAPHORE *operation_semaphore;
DICTIONARY *query_dictionary;
DICTIONARY *original_post_dictionary;
DRILLDOWN_PRIMARY *drilldown_primary;
LIST *drilldown_one_to_many_list;
LIST *drilldown_many_to_one_list;
DRILLDOWN_DOCUMENT *drilldown_document;
```

DRILLDOWN_INPUT (1)

```
/* Usage */
DRILLDOWN_INPUT *drilldown_input_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *drilldown_base_folder_name,
    char *target_frame,
    char *drilldown_primary_data_list_string,
    char *data_directory,
    DICTIONARY *original_post_dictionary );

/* Process */
DRILLDOWN_INPUT *drilldown_input_calloc(
    void );

FOLDER *folder =
    folder_fetch(
        drilldown_base_folder_name,
            /* Set to cache all folders for a role */
        role_name,
        (LIST *)0 /* exclude_attribute_name_list */,
        /* Sets folder_attribute_primary_list */
        /* Sets folder_attribute_primary_key_list */
        /* Sets folder_attribute_name_list */
        1 /* fetch_folder_attribute_list */,
        1 /* fetch_attribute */ );

    LIST *relation_mto1_list(
        drilldown_base_folder_name
            /* many_folder_name */,
        folder->folder_attribute_primary_key_list
            /* many_primary_key_list */ );

    LIST *relation_mto1_isa_list(
        (LIST *)0 /* mto1_isa_list Pass in null */,
        drilldown_base_folder_name
            /* many_folder_name */,
        folder->folder_attribute_primary_key_list
            /* many_primary_key_list */,
        0 /* not fetch_relation_one2m_list */,
        1 /* fetch_relation_mto1_list */ );

    LIST *relation_one2m_list(
        drilldown_base_folder_name
            /* one_folder_name */,
        folder->folder_attribute_primary_key_list
            /* one_primary_key_list */,
        1 /* include_isa_boolean */ );

    LIST *folder_attribute_append_isa_list(
        folder->folder_attribute_list,
        relation_mto1_isa_list() );
```



DRILLDOWN_INPUT (2)

```
/* Process (continued) */
boolean application_menu_horizontal_boolean(
    application_name );

boolean menu_horizontal_boolean(
    target_frame /* current_frame */,
    FRAMESET_PROMPT_FRAME,
    application_menu_horizontal_boolean() );

if ( menu_horizontal_boolean() )
{
    FOLDER_MENU *folder_menu =
        folder_menu_fetch(
            application_name,
            session_key,
            role_name,
            data_directory,
            0 /* not folder_menu_remove_boolean */ );

    MENU *menu =
        menu_new(
            application_name,
            session_key,
            login_name,
            role_name,
            1 /* application_menu_horizontal_boolean */,
            folder_menu->count_list );
}

DICTIONARY_SEPARATE_TABLE_EDIT *
    dictionary_separate =
        dictionary_separate_table_edit_new(
            original_post_dictionary,
            application_name,
            login_name,
            (LIST *)0 /* folder_attribute_date_name_list */,
            folder_attribute_append_isa_list() );

char *drilldown_input_post_table_edit_folder_name(
    const char *POST_TABLE_EDIT_FOLDER_LABEL,
    dictionary_separate->non_prefixed_dictionary );

LIST *drilldown_input_primary_attribute_data_list(
    ATTRIBUTE_MULTI_KEY_DELIMITER,
    drilldown_primary_data_list_string );

pid_t process_id = getpid();

/* Usage */
LIST *drilldown_input_primary_attribute_data_list(
    const char *ATTRIBUTE_MULTI_KEY_DELIMITER,
    char *drilldown_primary_data_list_string );

/* Process */
char *string_unescape_character(
    unescaped[],
    drilldown_primary_data_list_string,
    ATTRIBUTE_MULTI_KEY_DELIMITER );

LIST *attribute_data_list(
    ATTRIBUTE_MULTI_KEY_DELIMITER,
    unescaped );

/* Attributes */
FOLDER *folder;
LIST *relation_mto1_list;
LIST *relation_mto1_isa_list;
LIST *relation_one2m_list;
LIST *folder_attribute_append_isa_list;
boolean application_menu_horizontal_boolean;
boolean menu_horizontal_boolean;
FOLDER_MENU *folder_menu;
MENU *menu;
DICTIONARY_SEPARATE_TABLE_EDIT *
    dictionary_separate;
char *post_table_edit_folder_name;
LIST *primary_attribute_data_list;
pid_t process_id;
```

DICTIONARY_SEPARATE_TABLE_EDIT

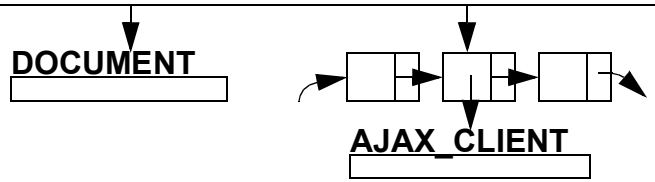
```
DICTIONARY *original_post_dictionary
DICTIONARY *sort_dictionary
DICTIONARY *drillthru_dictionary
DICTIONARY *no_display_dictionary
LIST *no_display_name_list
DICTIONARY *non_prefixed_dictiionary
DICTIONARY *query_dictionary
```

FOLDER_MENU

MENU

DRILLDOWN_DOCUMENT (1)

```
/* Usage */
DRILLDOWN_DOCUMENT *drilldown_document_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *drilldown_base_folder_name,
    char *update_results_string,
    char *update_error_string,
    MENU *drilldown_input->menu,
    char *folder_notepad,
    char *folder_javascript_filename,
    char *folder_post_change_javascript,
    LIST *relation_mto1_list,
    LIST *relation_one2m_list,
    DRILLDOWN_PRIMARY *drilldown_primary,
    LIST *drilldown_one_to_many_list );

/* Process */
DRILLDOWN_DOCUMENT *
    drilldown_document_calloc(
        void );

char *drilldown_document_title_string(
    char *drilldown_base_folder_name );

char *document_head_menu_setup_string(
    (menu) ? 1 : 0 /* menu_boolean */ ),

char *document_head_calendar_setup_string(
    1 /* folder_attribute_date_name_list_length */ );

LIST *drilldown_document_javascript_filename_list(
    folder_javascript_filename,
    relation_one2m_list,
    relation_mto1_list );

char *javascript_include_string(
     application_name,
     drilldown_document_javascript_filename_list() );

char *drilldown_document_onload_string(
     folder_post_change_javascript,
     relation_one2m_list,
     relation_mto1_list );

char *drilldown_document_results_string(
    char *update_results_string );

if ( update_error_string && *update_error_string )
{
    drilldown_document_results_string() = NULL;
}

DOCUMENT *document =
    document_new(
        application_name,
        application_title_string( application_name ),
        drilldown_document_title_string(),
        drilldown_document_results_string(),
        update_error_string /* sub_sub_title_string */,
        folder_notepad,
        drilldown_document_onload_string()
            /* onload_javascript_string */,
        (menu) ? menu->html : (char *)0,
        document_head_menu_setup_string(),
        document_head_calendar_setup_string(),
        javascript_include_string() );
        drilldown_document_onload_string()

LIST *drilldown_document_ajax_client_list(
    drilldown_primary,
    drilldown_one_to_many_list );

if ( list_length(
        drilldown_document_ajax_client_list() ) )
{
    char *ajax_client_list_javascript(
        session_key,
        login_name,
        role_name,
        drilldown_document_ajax_client_list() );
}
else
{
    ajax_client_list_javascript = "";
}

/* Attributes */
char *title_string;
char *document_head_menu_setup_string;
char *document_head_calendar_setup_string;
LIST *javascript_filename_list;
char *javascript_include_string;
char *onload_string;
DOCUMENT *document;
LIST *ajax_client_list;
char *ajax_client_list_javascript;
```

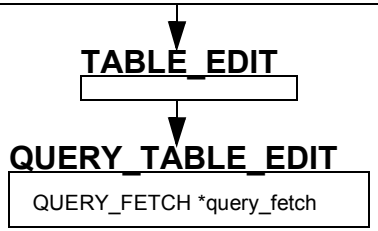


DRILLDOWN_DOCUMENT (2)

```
/* Usage */
LIST *drilldown_document_javascript_filename_list(
    char *folder_javascript_filename,
    LIST *relation_one2m_list(),
    LIST *relation_mto1_list() );

/* Process */
LIST *list = list_new();

list_set( list, “table_edit.js” );

if ( folder_javascript_filename )
{
    list_set( list, folder_javascript_filename );
}

for RELATION_ONE2M *relation_one2m in
    relation_one2m_list
{
    list_set(
        list,
        relation_one2m->
            many_folder->
            javascript_filename );
}

for RELATION_MTO1 *relation_mto1 in
    relation_mto1_list
{
    list_set(
        list,
        relation_mto1->
            one_folder->
            javascript_filename );
}

/* Usage */
char *drilldown_document_onload_string(
     char *folder_post_change_javascript,
     LIST *relation_one2m_list(),
     LIST *relation_mto1_list() );

/* Process */
char onload_string[];
char *ptr = onload_string;

if ( folder_post_change_javascript )
{
    char *javascript_replace(
        post_change_javascript,
        APPASERVER_UPDATE_STATE,
        -1 /* row_number */ );

    ptr += sprintf( ptr,
        “%s”,
        folder_post_change_javascript );
}

for RELATION_ONE2M *relation_one2m in
    relation_one2m_list
{
    if ( relation_one2m->
            many_folder->
            post_change_javascript )
    {
        if ( ptr != onload_string )
            ptr += sprintf( ptr, “ && “ );

        char *javascript_replace(
            relation_one2m->
                many_folder->
                post_change_javascript,
            APPASERVER_UPDATE_STATE,
            -1 /* row_number */ );

        ptr += sprintf( ptr,
            “%s”,
            javascript_replace() );
    }
}

for RELATION_MTO1 *relation_mto1 in
    relation_mto1_list
{
    if ( relation_mto1->
            one_folder->
            post_change_javascript )
    {
        if ( ptr != onload_string )
            ptr += sprintf( ptr, “ && “ );

        char *javascript_replace(
            relation_mto1->
                one_folder->
                post_change_javascript,
            APPASERVER_UPDATE_STATE,
            -1 /* row_number */ );

        ptr += sprintf( ptr,
            “%s”,
            javascript_replace() );
    }
}

/* Usage */
LIST *drilldown_document_ajax_client_list(
    DRILLDOWN_PRIMARY *drilldown_primary,
    LIST *drilldown_one_to_many_list );

/* Process */
LIST *ajax_client_list = list_new();

LIST *row_security_ajax_client_list(
    drilldown_primary->
        table_edit->
        row_security );

list_set_list(
    ajax_client_list,
    row_security_ajax_client_list() );

for DRILLDOWN_ONE_TO_MANY *
    drilldown_one_to_many in
        drilldown_one_to_many_list
{
    LIST *row_security_ajax_client_list(
        drilldown_one_to_many->
            table_edit->
            row_security );

        list_set_list(
            ajax_client_list,
            row_security_ajax_client_list() );
}
```

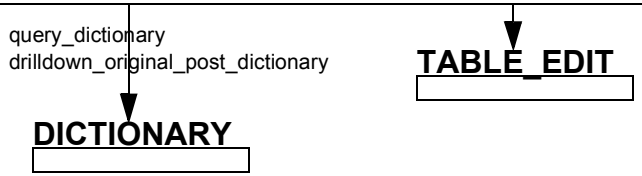
DRILLDOWN_PRIMARY

```
/* Usage */
DRILLDOWN_PRIMARY *drilldown_primary_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *drilldown_base_folder_name,
    char *target_frame,
    char *drilldown_primary_data_list_string,
    char *data_directory,
    pid_t drilldown_input->process_id,
    DICTIONARY *drilldown_original_post_dictionary );

/* Process */
DRILLDOWN_PRIMARY *drilldown_primary_calloc(
    void );

TABLE_EDIT *table_edit =
    table_edit_new(
        application_name,
        session_key,
        login_name,
        role_name,
        drilldown_base_folder_name,
        target_frame,
        process_id,
        (char *)0 /* results_string */,
        (char *)0 /* error_string */,
        0 /* not application_menu_horizontal_boolean */,
        drilldown_original_post_dictionary,
        data_directory,
        drilldown_base_folder_name,
        drilldown_primary_data_list_string,
        (LIST *)0 /* drilldown_relation_foreign_key_list */,
        0 /* not viewonly_boolean */,
        0 /* not omit_delete_boolean */ );

if ( !table_edit ) exit( 1 );

/* Attributes */
TABLE_EDIT *table_edit;
```



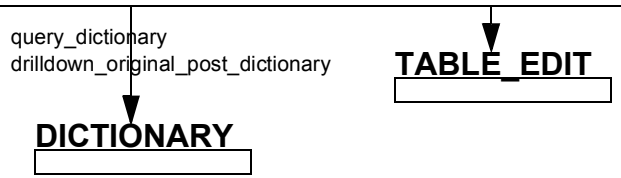
DRILLDOWN_ONE_TO_MANY

```
/* Usage */
DRILLDOWN_ONE_TO_MANY *
   drilldown_one_to_many_new(
      char *application_name,
      char *session_key,
      char *login_name,
      char *role_name,
      char *drilldown_base_folder_name,
      char *target_frame,
      char *drilldown_primary_data_list_string,
      char *data_directory,
      pid_t process_id,
      DICTIONARY *no_display_dictionary,
      DICTIONARY *sort_dictionary,
      LIST *primary_attribute_data_list,
      char *relation_many_folder_name,
      LIST *relation_foreign_key_list );

/* Process */
DRILLDOWN_ONE_TO_MANY *
   drilldown_one_to_many_calloc(
      void );

DICTIONARY *
   drilldown_one_to_many_query_dictionary(
      QUERY_RELATION_OPERATOR_PREFIX,
      QUERY_EQUAL,
      sort_dictionary,
      primary_attribute_data_list,
      relation_foreign_key_list );

DICTIONARY *drilldown_original_post_dictionary(
   DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
   DICTIONARY_SEPARATE_SORT_PREFIX,
   DICTIONARY_SEPARATE_QUERY_PREFIX,
   no_display_dictionary,
   sort_dictionary,
   drilldown_one_to_many_query_dictionary() );

TABLE_EDIT *table_edit =
   table_edit_new(
      application_name,
      session_key,
      login_name,
      role_name,
      relation_many_folder_name,
      target_frame,
      process_id,
      (char *)0 /* results_string */,
      (char *)0 /* error_string */,
      0 /* not application_menu_horizontal_boolean */,
      drilldown_original_post_dictionary(),
      data_directory,
      drilldown_base_folder_name,
      drilldown_primary_data_list_string,
      relation_foreign_key_list,
      0 /* not viewonly_boolean */,
      0 /* not omit_delete_boolean */ );

if (    !table_edit
||      !table_edit->query_row_list_length )
{
   return  NULL;
}

/* Usage */
DICTIONARY *
   drilldown_one_to_many_query_dictionary(
      const char *QUERY_RELATION_OPERATOR_PREFIX,
      const char *QUERY_EQUAL,
      LIST *primary_attribute_data_list,
      LIST *relation_foreign_key_list );

/* Process */
DICTIONARY *attribute_query_dictionary(
   QUERY_RELATION_OPERATOR_PREFIX,
   QUERY_EQUAL,
   relation_foreign_key_list,
   primary_attribute_data_list );

/* Attributes */
char *relation_many_folder_name;
DICTIONARY *query_dictionary;
DICTIONARY *drilldown_original_post_dictionary;
TABLE_EDIT *table_edit;
```



DRILLDOWN_MANY_TO_ONE

```
/* Usage */
DRILLDOWN_MANY_TO_ONE *
   drilldown_many_to_one_new(
      char *application_name,
      char *session_key,
      char *login_name,
      char *role_name,
      char *drilldown_base_folder_name,
      char *target_frame,
      char *drilldown_primary_data_list_string,
      char *data_directory,
      pid_t drilldown_input->process_id,
      DICTIONARY *sort_dictionary,
      char *relation_mto1->one_folder_name,
      LIST *relation_mto1->relation_foreign_key_list,
      LIST *relation_mto1->relation_translate_list,
      LIST *one_folder_primary_key_list,
      DICTIONARY *query_fetch_dictionary() );

/* Process */
DRILLDOWN_MANY_TO_ONE *
   drilldown_many_to_one_calloc(
      void );

LIST *drilldown_many_to_one_attribute_data_list(
   relation_translate_list,
   one_folder_primary_key_list,
   query_fetch_dictionary );

if ( !drilldown_many_to_one_attribute_data_list() )
      return NULL;

DICTIONARY *
   drilldown_many_to_one_query_dictionary(
      QUERY_RELATION_OPERATOR_PREFIX,
      QUERY_EQUAL,
      one_folder_primary_key_list,
      drilldown_many_to_one_attribute_data_list() );

DICTIONARY *drilldown_original_post_dictionary(
   DICTIONARY_SEPARATE_NO_DISPLAY_PREFIX,
   DICTIONARY_SEPARATE_SORT_PREFIX,
   DICTIONARY_SEPARATE_QUERY_PREFIX,
   (DICTIONARY *)0 /* no_display_dictionary */,
   sort_dictionary,
   drilldown_many_to_one_query_dictionary() );

TABLE_EDIT *table_edit =
   table_edit_new(
      application_name,
      session_key,
      login_name,
      role_name,
      one_folder_name,
      target_frame,
      process_id,
      relation_foreign_key_list,
      (char *)0 /* results_string */,
      (char *)0 /* error_string */,
      0 /* not application_menu_horizontal_boolean */,
      drilldown_original_post_dictionary(),
      data_directory,
      drilldown_base_folder_name,
      drilldown_primary_data_list_string,
      relation_foreign_key_list,
      0 /* not viewonly_boolean */,
      1 /* omit_delete_boolean */ );

/* If lookup permission isn't set. */
if ( !table_edit ) return NULL;

/* Usage */
LIST *drilldown_many_to_one_attribute_data_list(
   LIST *relation_translate_list,
   LIST *one_folder_primary_key_list,
   DICTIONARY *query_fetch_dictionary )

/* Process */
LIST *attribute_data_list = list_new();

for char *primary_key in one_folder_primary_key_list
{
   char *relation_translate_foreign_key(
      relation_translate_list,
      primary_key );

   char *dictionary_get(
      relation_translate_foreign_key(),
      query_fetch_dictionary );

   if ( !dictionary_get() ) return NULL;

   list_set(
      attribute_data_list,
      dictionary_get() );
}

/* Usage */
DICTIONARY *
   drilldown_many_to_one_query_dictionary(
      const char *QUERY_RELATION_OPERATOR_PREFIX,
      const char *QUERY_EQUAL,
      LIST *one_folder_primary_key_list,
      LIST *drilldown_many_to_one_attribute_data_list );

/* Process */
DICTIONARY *attribute_query_dictionary(
   QUERY_RELATION_OPERATOR_PREFIX,
   QUERY_EQUAL,
   one_folder_primary_key_list,
   drilldown_many_to_one_attribute_data_list );

/* Attributes */
char *one_folder_name;
LIST *attribute_data_list;
DICTIONARY *query_dictionary;
DICTIONARY *drilldown_original_post_dictionary;
TABLE_EDIT *table_edit;
```



SPOOL

```
/* Usage */

/* Output in spool_list(spool_new()->output_filespecification) */
SPOOL *spool_new(
    char *system_string,
    boolean capture_stderr_boolean );

/* Process */
SPOOL *spool_calloc(
    void );

char *spool_output_filespecification(
    pid_t getpid() /* process_id */ );

char *spool_create_system_string(
    char *spool_output_filespecification() );

system( spool_create_system_string() );

char *spool_output_system_string(
    char *system_string,
    boolean capture_stderr_boolean,
    char *spool_output_filespecification() );

FILE *spool_output_pipe(
    spool_output_system_string() );

/* Usage */
FILE *spool_output_pipe(
    char *spool_output_system_string() );

/* Process */
FILE *appaserver_output_pipe(
    spool_output_system_string );

/* Usage */
LIST *spool_list(
    char *spool->output_filespecification );

/* Process */
char *appaserver_input_file(
    char *output_filespecification );

LIST *list_file_fetch(
    appaserver_input_file() );

fclose( appaserver_input_file() );

char *spool_remove_system_string(
    char *spool_output_filespecification() );

system( spool_remove_system_string() );

/* Usage */
char *spool_pipe(
    char *system_string,
    boolean capture_stderr_boolean );

/* Process */
SPOOL *spool_new(
    system_string,
    capture_stderr_boolean );

pclose( spool_new()->output_pipe );
LIST *spool_list( spool_new()->output_filespecification );
int list_length( spool_list() );

if ( !list_length() ) return NULL;

(char *)list_first( spool_list() );

/* Usage */
char *spool_fetch(
    char *system_string,
    boolean capture_stderr_boolean );

/* Process */
char *spool_pipe(
    system_string,
    capture_stderr_boolean );

/* Usage */
char *spool_data_string(
    char *system_string,
    char *data_string );

/* Process */
SPOOL *spool_new(
    system_string,
    1 /* capture stderr_boolean */ );

fprintf(
    spool_new()->output_pipe,
    "%s\n",
    data_string );

pclose( spool_new()->output_pipe );

LIST *spool_list( spool_new()->output_filespecification );
int list_length( spool_list() );

if ( !list_length() ) return NULL;

(char *)list_first( spool_list() );

/* Driver */
for char *string
{
    fprintf(
        spool->output_pipe,
        "%s\n",
        string );
}

pclose( spool->output_pipe );

LIST *spool_list(
    spool->output_filespecification );

/* Attributes */
char *output_filespecification;
char *create_system_string;
char *output_system_string;
FILE *output_pipe;
```

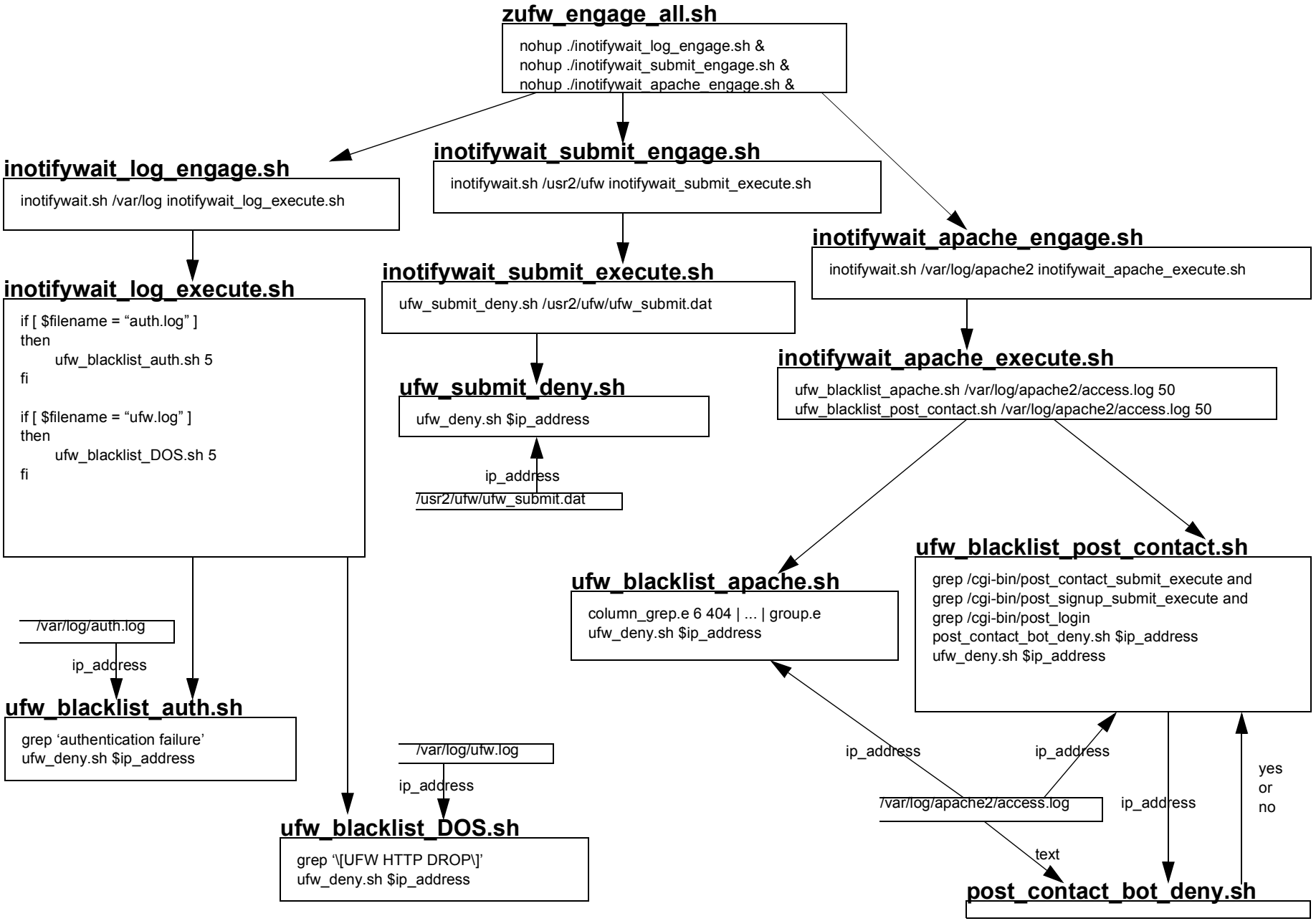
PREUPDATE_CHANGE (1)

```
/* Usage */
PREUPDATE_CHANGE *preupdate_change_new(
    const char *APPASERVER_INSERT_STATE,
    const char *APPASERVER_PREDELETE_STATE,
    char *state,
    char *preupdate_datum,
    char *postupdate_datum,
    const char *preupdate_placeholder_name );

/* Process */
PREUPDATE_CHANGE *preupdate_change_calloc(
    void );

char *preupdate_change_preupdate_datum(
    preupdate_placeholder_name,
    preupdate_datum,
    postupdate_datum );

enum preupdate_change_state
    preupdate_change_state_evaluate(
        APPASERVER_INSERT_STATE,
        APPASERVER_PREDELETE_STATE,
        state,
        preupdate_change_preupdate_datum(),
        postupdate_datum );

boolean preupdate_change_no_change_boolean(
    enum preupdate_change_state
        preupdate_change_state_evaluate();

char *preupdate_change_prior_datum(
    preupdate_change_preupdate_datum()
        /* preupdate_attribute_datum */,
    preupdate_change_state_evaluate() );

char *preupdate_change_new_datum(
    postupdate_datum
        /* attribute_datum */,
    preupdate_change_state_evaluate() );

/* Attributes */
enum preupdate_change_state state_evaluate;
boolean no_change_boolean;
char *prior_datum;
char *new_datum;
```

enum preupdate_change_state

no_change_null
from_null_to_something
from_something_to_null
from_something_to_something_else
no_change_something

PREUPDATE_CHANGE (2)

```
/* Usage */
enum preupdate_change_state
    preupdate_change_state_evaluate(
        const char *APPASERVER_INSERT_STATE,
        const char *APPASERVER_PREDELETE_STATE,
        char *state,
       char *preupdate_datum,
       char *postupdate_datum );

/* Process */
if (    !preupdate_datum
||      strcmp( preupdate_datum, "select" ) == 0 )
{
    preupdate_datum = "";
}

if (    !postupdate_datum ||
        strcmp( postupdate_datum, "/" ) == 0 )
{
    postupdate_datum = "";
}

if ( strcmp(
        state,
        APPASERVER_INSERT_STATE ) == 0 )
{
    if ( *postupdate_datum )
            state_evaluate = from_null_to_something;
    else
            state_evaluate = no_change_null;
}
else
if ( strcmp(
        state,
        APPASERVER_PREDELETE_STATE ) == 0 )
{
    if ( *postupdate_datum )
            state_evaluate = from_something_to_null;
    else
            state_evaluate = no_change_null;
}
else
if ( !*preupdate_datum && !*postupdate_datum )
{
    state_evaluate = no_change_null;
}
else
if ( !*postupdate_datum )
{
        state_evaluate = from_something_to_null;
}
else
if ( !*preupdate_datum )
{
        state_evaluate = from_null_to_something;
}
else
if ( strcmp(
        preupdate_datum,
        postupdate_datum ) == 0 )
{
   state_evaluate = no_change_something;
}
else
{
        state_evaluate =
           from_something_to_something_else;
}

/* Usage */
char *preupdate_change_prior_datum(
    char *preupdate_attribute_datum,
    enum preupdate_change_state
        preupdate_change_state );

/* Process */
if (    preupdate_change_state ==
            from_null_to_something
||      preupdate_change_state ==
            no_change_null )
{
    prior_datum = null;
}
else
if (    preupdate_change_state ==
            from_something_to_null
||      preupdate_change_state ==
            from_something_to_something_else
||      preupdate_change_state ==
            no_change_something )
{
    prior_datum = preupdate_attribute_datum;
}

/* Usage */
char *preupdate_change_new_datum(
    char *attribute_datum,
    enum preupdate_change_state
        preupdate_change_state );

/* Process */
if (    preupdate_change_state ==
            from_something_to_null
||      preupdate_change_state ==
            no_change_null )
{
    datum = null;
}
else
if (    preupdate_change_state ==
            from_null_to_something
||      preupdate_change_state ==
            from_something_to_something_else
||      preupdate_change_state ==
            no_change_something )
{
    datum = attribute_datum;
}

/* Usage */
char *preupdate_change_preupdate_datum(
    const char *preupdate_placeholder_name,
    char *preupdate_datum,
    char *postupdate_datum );

/* Process */
if ( string_strcmp(
        preupdate_placeholder_name,
        preupdate_datum ) == 0 )
{
    preupdate_datum = postupdate_datum;
}

/* Usage */
char *preupdate_change_state_display(
    enum preupdate_change_state state_evaluate );

/* Process */
```

UFW Overview