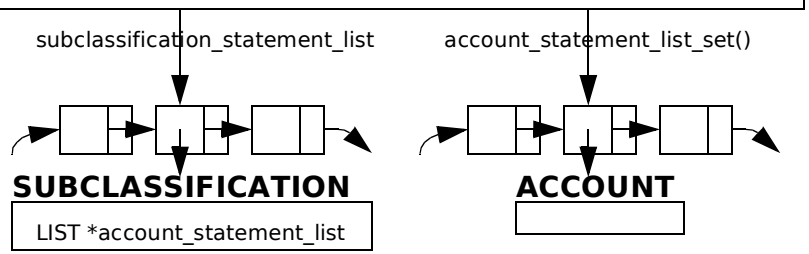
PredictBooks Algorithmic Application Programming Interface

ELEMENT (1)

```
/* Usage */
ELEMENT *element_fetch(
    char *element_name );

/* Process */
static LIST *element_list();

ELEMENT *element_seek(
    element_name,
    element_list() );

/* Usage */
LIST *element_list( void );

/* Process */
char *element_system_string(
    ELEMENT_SELECT,
    ELEMENT_TABLE,
    (char *)0 /* where */ );

LIST *element_system_list(
    element_system_string() );
```

```
/* Usage */
LIST *element_system_list(
    char *element_system_string() );

/* Process */
FILE *element_pipe(
    char *element_system_string() );

for input[] in string_input( element_pipe() )
{
    list_set(
        list,
        element_parse( input ) );
}

pclose( element_pipe() );

/* Usage */
ELEMENT *element_parse(
    char *input );
```

```
/* Process */
ELEMENT *element_new(
    element_name );

/* Usage */
ELEMENT *element_new(
    char *element_name );

/* Process */
ELEMENT *element_calloc(
    void );

/* Attributes */
char *element_name;
boolean accumulate_debit;

/* Set externally */
LIST *subclassification_statement_list;
LIST *account_statement_list;
double sum;
int percent_of_asset;
int percent_of_income;
```



ELEMENT (2)

```
/* Usage */
LIST *element_statement_list(
    LIST *element_name_list,
    char *end_date_time_string,
    boolean fetch_subclassification_list,
    boolean fetch_account_list,
    boolean fetch_journal_latest,
    boolean fetch_transaction );

/* Process */
for char *element_name in element_name_list
{
    list_set(
        element_list,
        element_statement_fetch(
            element_name,
            end_date_time_string,
            fetch_subclassification_list,
            fetch_account_list,
            fetch_journal_latest,
            fetch_transaction ) );
}

/* Usage */
ELEMENT *element_statement_fetch(
    char *element_name,
    char *end_date_time_string,
    boolean fetch_subclassification_list,
    boolean fetch_account_list,
    boolean fetch_journal_latest,
    boolean fetch_transaction );

/* Process */
char *element_primary_where(
    char *element_name );

char *element_system_string(
    char *ELEMENT_SELECT,
    char *ELEMENT_TABLE,
    char *element_primary_where() /* where */ );

FILE *element_pipe(
    element_system_string() );

char *string_input( input[], element_pipe() );

ELEMENT *element_statement_parse(
    string_input(),
    end_date_time_string,
    fetch_subclassification_list,
    fetch_account_list,
    fetch_journal_latest,
    fetch_transaction );

pclose( element_pipe() );

/* Usage */
ELEMENT *element_statement_parse(
    char *string_input(),
    char *end_date_time_string,
    boolean fetch_subclassification_list,
    boolean fetch_account_list,
    boolean fetch_journal_latest,
    boolean fetch_transaction );

/* Process */
ELEMENT *element =
    element_parse(
        string_input );

if ( fetch_subclassification_list )
{
    char *element_primary_where(
        element_name );

    LIST *subclassification_statement_list(
        element_primary_where(),
        end_date_time_string,
        fetch_account_list,
        fetch_journal_latest,
        fetch_transaction );
}
```

ELEMENT (3)

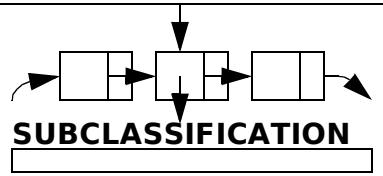
```
/* Usage */
void element_list_sum_set(
    LIST *element_statement_list );

/* Process */
for ELEMENT *element in element_statement_list
{
    element->sum =
       element_sum(
          element );
}

/* Usage */
double element_sum( ELEMENT *element );

/* Process */
double subclassification_list_sum_set(
    /* Sets subclassification->sum */
    element->subclassification_statement_list );

/* Usage */
void element_list_account_statement_list_set(
   LIST *element_statement_list /* in/out */ );

/* Process */
for ELEMENT *element in element_statement_list
{
    if ( element->sum )
    {
       element->account_statement_list =
          element_account_statement_list(
             element->subclassification_statement_list,
             element->account_statement_list );
    }
}

/* Usage */
LIST *element_account_statement_list(
    LIST *subclassification_statement_list,
    LIST *account_statement_list );

/* Process */
if ( list_length( subclassification_statement_list ) )
{
    LIST *account_statement_list =
       LIST *subclassification_account_statement_list(
          subclassification_statement_list );
}
else
if ( list_length( account_statement_list ) )
{
    LIST *account_statement_list =
       LIST *account_balance_sort_list(
          account_statement_list );
}

/* Usage */
void element_delta_prior_percent_set(
    ELEMENT *prior_element /* in/out */,
    ELEMENT *current_element );

/* Process */
int prior_element->delta_prior_percent =
    int float_delta_prior_percent(
       prior_element->sum,
       current_element->sum );

void subclassification_list_delta_prior_percent_set(
    prior_element->
       subclassification_statement_list
          /* prior_subclassification_list in/out */,
    current_element->
       subclassification_statement_list
          /* current_subclassification_list */ );

/* Usage */
void element_account_transaction_count_set(
   LIST *element_statement_list,
   char *transaction_begin_date_string(),
   char *end_date_time_string );

/* Process */
for ELEMENT *element in element_statement_list
{
   subclassification_account_transaction_count_set(
      element->subclassification_statement_list,
      transaction_begin_date_string,
      end_date_time_string );
}
```

ELEMENT (4)

```
/* Usage */
void element_account_action_string_set(
    LIST *element_statement_list,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *transaction_begin_date_string(),
    char *end_date_time_string );

/* Process */
for ELEMENT *element in element_statement_list
{
    if ( element->sum )
    {
        void subclassification_account_action_string_set(
            element->subclassification_statement_list,
            application_name,
            session_key,
            login_name,
            role_name,
            transaction_begin_date_string,
            end_date_time_string );
    }
}

/* Usage */
void element_percent_of_asset_set(
    LIST *element_statement_list );

/* Process */
ELEMENT *element_asset =
    element_seek(
        ELEMENT_ASSET,
        element_statement_list );

for ELEMENT *element in element_statement_list
{
    if ( element->sum
    && !element_is_nominal( element->element_name ) )
    {
        int element->percent_of_asset =
            int float_percent_of_total(
                /* Set by statement_fetch()->
                    element_list_sum_set() */
                element->sum,
                element_asset->sum );

        subclassification_percent_of_asset_set(
            element->subclassification_statement_list,
            element_asset->sum );
    }
}

/* Usage */
void element_percent_of_income_set(
    LIST *element_statement_list );

/* Process */
ELEMENT *element_revenue =
    element_seek(
        ELEMENT_REVENUE,
        element_statement_list );

ELEMENT *element_gain =
    element_seek(
        ELEMENT_GAIN,
        element_statement_list );

double element_income(
    ELEMENT *element_revenue,
    ELEMENT *element_gain );

if ( element_income() )
{
    for ELEMENT *element in element_statement_list
    {
        if ( element_is_nominal( element->element_name )
        {
            int element->percent_of_income =
                int float_percent_of_total(
                    /* Set by statement_fetch()->
                        element_list_sum_set() */
                    element->sum,
                    element_income() );

            subclassification_percent_of_income_set(
                element->subclassification_statement_list,
                element_income() );
        }
    }
}

/* Usage */
double element_net_income(
    LIST *element_statement_list );

/* Process */
double revenue =
    element_seek_sum(
        ELEMENT_REVENUE,
        element_statement_list );

double gain =
    element_seek_sum(
        ELEMENT_GAIN,
        element_statement_list );

double expense =
    element_seek_sum(
        ELEMENT_EXPENSE,
        element_statement_list );

double loss =
    element_seek_sum(
        ELEMENT_LOSS,
        element_statement_list );

return (revenue + gain) - (expense + loss);

/* Public */
boolean element_is_nominal(
    char *element_name );

ELEMENT *element_seek(
    char *element_name,
    LIST *element_list );

double element_list_debit_sum(
    LIST *element_statement_list );

double element_list_credit_sum(
    LIST *element_statement_list );

double element_seek_sum(
    char *element_name,
    LIST *element_statement_list );
```

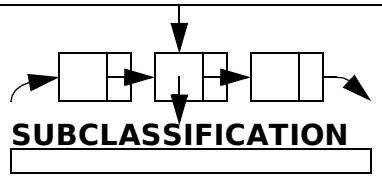
ELEMENT_FUND

```
/* Usage */
ELEMENT_FUND *element_fund_fetch(
    char *element_name,
    char *fund_name,
    char *begin_transaction_date_time,
    char *end_transaction_date_time,
    boolean fetch_subclassification_list,
    boolean fetch_account_list,
    boolean fetch_journal_list );

/* Process */
char *element_primary_where(
    element_name );

char *element_system_string(
    ELEMENT_SELECT,
    ELEMENT_TABLE,
    element_primary_where()
        /* where */ );

ELEMENT_FUND *element_fund_parse(
    char *input,
    char *fund_name,
    char *begin_transaction_date_time,
    char *end_transaction_date_time,
    boolean fetch_subclassification_list,
    boolean fetch_account_list,
    boolean fetch_journal_list );

ELEMENT_FUND *element_fund_new(
    char *element_name,
    char *fund_name );

ELEMENT_FUND *element_fund_calloc(
    void );

/* Attributes */
char *element_name;
char *fund_name;
boolean accumulate_debit;
LIST *subclassifiction_list;
```



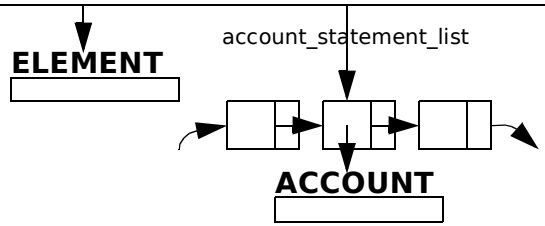
ELEMENT_PROGRAM

```
/* Usage */
ELEMENT_PROGRAM *element_program_fetch(
    char *element_name,
    char *program_name,
    char *begin_transaction_date_time,
    char *end_transaction_date_time,
    boolean fetch_subclassification_list,
    boolean fetch_account_list,
    boolean fetch_journal_list );

/* Process */
char *element_primary_where(
    element_name );

char *element_system_string(
    ELEMENT_SELECT,
    ELEMENT_TABLE,
    element_primary_where()
        /* where */ );

ELEMENT_PROGRAM *element_program_parse(
    char *input,
    char *program_name,
    char *begin_transaction_date_time,
    char *end_transaction_date_time,
    boolean fetch_subclassification_list,
    boolean fetch_account_list,
    boolean fetch_journal_list );

ELEMENT_PROGRAM *element_program_new(
    char *element_name,
    char *program_name );

ELEMENT_PROGRAM *element_program_calloc(
    void );

/* Attributes */
char *element_name;
char *program_name;
boolean accumulate_debit;
LIST *subclassification_list;
```



SUBCLASSIFICATION (1)

```
/* Usage */
SUBCLASSIFICATION *subclassification_fetch(
    char *subclassification_name,
    boolean fetch_element );

/* Process */
if ( fetch_element )
{
    static LIST *subclassification_list(
        "1 = 1" /* where */,
        1 /* fetch_element */ );

    SUBCLASSIFICATION *subclassification_seek(
        char *subclassification_name,
        LIST *subclassification_list() );
}
else
{
    char *subclassification_primary_where(
        char *subclassification_name );

    char *subclassification_system_string(
        SUBCLASSIFICATION_SELECT,
        SUBCLASSIFICATION_TABLE,
        subclassification_primary_where() /* where */ );

    FILE *subclassification_pipe(
        subcalssification_system_string() );

    SUBCLASSIFICATION *subclassification_parse(
        string_input(
            input[],
            subclassification_pipe() ),
        0 /* not fetch_element */ );

    pclose( subclassification_pipe() );
}

/* Usage */
SUBCLASSIFICATION *subclassification_parse(
    char *input,
    boolean fetch_element );

/* Process */
SUBCLASSIFICATION *subclassification_new(
    char *subclassification_name );

SUBCLASSIFICATION *subclassification_calloc( void );

if ( fetch_element )
{
    ELEMENT *element =
        element_fetch(
            subclassification->element_name );
}

/* Attributes */
char *subclassification_name;
char *element_name;
int display_order;
ELEMENT *element;

/* Set externally */
LIST *account_statement_list;
double sum;
int percent_of_asset;
int percent_of_income;
```



SUBCLASSIFICATION (2)

```
/* Usage */
LIST *subclassification_statement_list(
    char *element_primary_where(),
    char *end_date_time_string,
    boolean fetch_account_list,
    boolean fetch_journal_latest,
    boolean fetch_transaction );

/* Process */
char *subclassification_system_string(
    char *SUBCLASSIFICATION_SELECT,
    char *SUBCLASSIFICATION_TABLE,
    char *where );

FILE *subclassification_pipe(
    char *subclassification_system_string() );

for char *input in
    string_input( input[], subclassification_pipe() )
{
    list_set(
        list,
        subclassification_statement_parse(
            input,
            end_date_time_string,
            fetch_account_list,
            fetch_journal_latest,
            fetch_transaction ) );
}

pclose( subclassification_pipe() );

/* Usage */
SUBCLASSIFICATION *
    subclassification_statement_parse(
        char *input,
        char *end_date_time_string,
        boolean fetch_account_list,
        boolean fetch_journal_latest,
        boolean fetch_transaction );

/* Process */
SUBCLASSIFICATION *subclassification =
    SUBCLASSIFICATION *subclassification_parse(
        input,
        0 /* not fetch_element */ );

if ( fetch_account_list )
{
    LIST *account_statement_list(
        subclassification_primary_where(
            subclassification_name ),
        end_date_time_string,
        0 /* not fetch_subclassification */,
        0 /* not fetch_element */,
        fetch_journal_latest,
        fetch_transaction );

    LIST *account_statement_list =
        LIST *account_balance_sort_list(
            account_statement_list );
}

/* Usage */
LIST *subclassification_account_statement_list(
    LIST *subclassification_statement_list() );

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

for SUBCLASSIFICATION *subclassification in
    subclassification_statement_list
{
    for ACCOUNT *account in
        subclassification->account_statement_list
    {
        list_set_order(
            account_statement_list /* in/out */,
            account,
            account_balance_compare_function );
    }
}
```

SUBCLASSIFICATION (3)

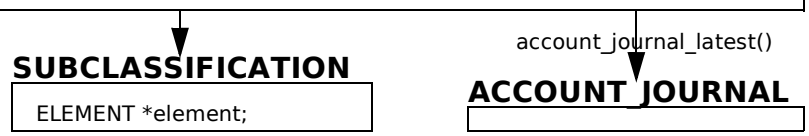
```
/* Usage */
LIST *subclassification_list(
    char *where,
    boolean fetch_element );

/* Process */
char *subclassification_system_string(
    SUBCLASSIFICATION_SELECT,
    SUBCLASSIFICATION_TABLE,
    where );

LIST *subclassification_system_list(
    subclassification_system_string(),
    fetch_element );

/* Usage */
LIST *subclassification_system_list(
    char *subclassification_system_string(),
    boolean fetch_element );

/* Process */
FILE *subclassification_pipe(
    subclassification_system_string() );

for char input[] in string_input( subclassification_pipe() )
{
    list_set(
        list,
        subclassification_parse(
            input,
            fetch_element ) );
}

pclose( subclassification_pipe() );

/* Usage */
void subclassification_list_sum_set(
    /* Sets subclassification->sum */
    LIST *subclassification_statement_list() );

/* Process */
for SUBCLASSIFICATION *subclassification in
    subclassification_statement_list
{
    subclassification->sum =
        account_list_sum(
            subclassification->
                account_statement_list );
    list_sum += subclassification->sum;
}

/* Usage */
void subclassification_list_delta_prior_percent_set(
    LIST *prior_subclassification_list
        /* in/out */,
    LIST *current_subclassification_list );

/* Process */
for SUBCLASSIFICATION *current_subclassifiation in
    current_subclassification_list
{
    SUBCLASSIFICATION *prior_subclassification =
        subclassification_seek(
            current_subclassification->
                subclassification_name,
            prior_subclassification_list
                /* subclassification_list */ );

    void subclassification_delta_prior_percent_set(
        prior_subclassification /* in/out */,
        current_subclassification );
}

/* Usage */
void subclassification_delta_prior_percent_set(
    SUBCLASSIFICATION *prior_subclassification
        /* in/out */,
    SUBCLASSIFICATION *current_subclassification );

/* Process */
int prior_subclassification->delta_prior_percent =
     int float_delta_prior_percent(
         prior_subclassification->sum,
         current_subclassification->sum );

void account_list_delta_prior_percent_set(
    prior_subclassification->account_statement_list
        /* prior_account_list in/out */,
    current_subclassification->account_statement_list
        /* current_account_list */ );

/* Usage */
void subclassification_account_statement_list(
    LIST *subclassification_statement_list );

/* Process */
for SUBCLASSIFICATION *subclassification in
    subclassification_statement_list
{
    list_set_list(
        list,
        subclassification->account_statement_list );
}

/* Usage */
void subclassification_account_transaction_count_set(
        LIST *subclassification_statement_list,
        char *transaction_begin_date_string(),
        char *end_date_time_string );

/* Process */
for SUBCLASSIFICATION *subclassification in
    subclassification_statement_list
{
    void account_transaction_count_set(
        subclassification->account_statement_list,
        transaction_begin_date_string,
        end_date_time_string );
}

/* Usage */
void subclassification_account_action_string_set(
    LIST *subclassification_statement_list,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *transaction_begin_date_string(),
    char *end_date_time_string );

/* Process */
for SUBCLASSIFICATION *subclassification in
    subclassification_statement_list
{
    void account_list_action_string_set(
        subclassification->account_statement_list,
        application_name,
        session_key,
        login_name,
        role_name,
        transaction_begin_date_string,
        end_date_time_string );
}
```

SUBCLASSIFICATION (4)

```
/* Usage */
void subclassification_percent_of_asset_set(
    LIST *subclassification_statement_list,
    double element_asset_sum );

/* Process */
for SUBCLASSIFICATION *subclassification in
    subclassification_statement_list
{
    int subclassification->percent_of_asset =
        int float_percent_of_total(
            /* Set by statement_fetch()->
               element_list_sum_set() */
            subclassification->sum,
            element_asset_sum );

    void account_percent_of_asset_set(
        subclassification->account_statement_list,
        element_asset_sum );
}

/* Usage */
void subclassification_percent_of_income_set(
    LIST *subclassification_statement_list,
    double element_income() );

/* Process */
for SUBCLASSIFICATION *subclassification in
    subclassification_statement_list
{
    int subclassification->percent_of_income =
        int float_percent_of_total(
            /* Set by statement_fetch()->
               element_list_sum_set() */
            subclassification->sum,
            element_income );

    void account_percent_of_income_set(
        subclassification->account_statement_list,
        element_income );
}

/* Usage */
SUBCLASSIFICATION *
    subclassification_element_list_seek(
        char *subclassification_name,
        LIST *element_statement_list );

/* Process */
for ELEMENT *element in element_statement_list
{
    SUBCLASSIFICATION *
        subclassification_seek(
            subclassification_name,
            element->subclassification_statement_list );
}

/* Public */
double subclassification_list_debit_sum(
    LIST *subclassification_statement_list,
    boolean element_accumulate_debit );

double subclassification_list_credit_sum(
    LIST *subclassification_statement_list,
    boolean element_accumulate_debit );

boolean subclassification_current_liability_boolean(
    char *SUBCLASSIFICATION_CURRENT_LIABILITY,
    char *subclassification_name );

boolean subclassification_receivable_boolean(
    char *SUBCLASSIFICATION_RECEIVABLE,
    char *subclassification_name );
```

ACCOUNT (1)

```
/* Usage */
LIST *account_statement_list(
    char *subclassification_primary_where(),
    char *end_date_time_string,
    boolean fetch_subclassification,
    boolean fetch_element,
    boolean fetch_journal_latest,
    boolean fetch_transaction );

/* Process */
boolean account_chart_account_boolean(
    ACCOUNT_TABLE,
    ACCOUNT_CHART_ACCOUNT_NUMBER );

char *account_select_string(
    ACCOUNT_SELECT,
    ACCOUNT_CHART_ACCOUNT_NUMBER,
    account_chart_account_boolean() );

char *account_system_string(
    account_select_string(),
    ACCOUNT_TABLE,
    subclassification_primary_where
       /* where */ );

FILE *account_pipe( char *account_system_string() );

for input[] in string_input( account_pipe() )
{
    list_set(
       statement_list,
       account_statement_parse(
          input,
          end_date_time_string,
          account_chart_account_boolean(),
          fetch_subclassification,
          fetch_element,
          fetch_journal_latest,
          fetch_transaction ) );
}

pclose( account_pipe() );

/* Usage */
ACCOUNT *account_statement_parse(
    char *input,
    char *end_date_time_string,
    boolean account_chart_account_boolean,
    boolean fetch_subclassification,
    boolean fetch_element,
    boolean fetch_journal_latest,
    boolean fetch_transaction );

/* Process */
ACCOUNT *account =
    account_parse(
       input,
       account_chart_account_boolean,
       fetch_subclassification,
       fetch_element );

if ( fetch_journal_latest )
{
    ACCOUNT_JOURNAL *account_journal_latest(
       account_name,
       end_date_time_string,
       fetch_transaction );
}

/* Usage */
ACCOUNT *account_key_fetch(
    const char *hard_coded_account_key,
    const char *calling_function_name );

/* Process */
char *account_hard_coded_account_name(
    hard_coded_account_key,
    0 /* not warning_only */,
    calling_function_name );

ACCOUNT *account_fetch(
    account_hard_coded_account_name(),
    1 /* fetch_subclassification */,
    1 /* fetch_element */ );

/* Attributes */
char *account_name;
char *subclassification_name;
char *hard_coded_account_key;
char *chart_account_number;
int annual_budget;
SUBCLASSIFICATION *subclassification;

/* Set externally */
ACCOUNT_JOURNAL *account_journal_latest;
int transaction_count;
double balance;
double payment_amount;
int percent_of_asset;
int percent_of_income;
int delta_prior_percent;
LIST *liability_journal_list;
double liability_due;
char *action_string;
```



ACCOUNT (2)

```
/* Usage */
ACCOUNT *account_fetch(
    char *account_name,
    boolean fetch_subclassification,
    boolean fetch_element );

/* Process */
if ( fetch_subclassification && fetch_element )
{
    static LIST *account_list(
        "1 = 1" /* where */,
        1 /* fetch_subclassification */,
        1 /* fetch_element */ );

    ACCOUNT *account_seek(
        account_name,
        account_list() );
}
else
{
    boolean account_chart_account_boolean(
        ACCOUNT_TABLE,
        ACCOUNT_CHART_ACCOUNT_NUMBER );

    char *account_select_string(
        ACCOUNT_SELECT,
        ACCOUNT_CHART_ACCOUNT_NUMBER,
        account_chart_account_boolean() );

    char *account_primary_where(
        char *account_name );

    char *account_system_string(
        account_select_string(),
        ACCOUNT_TABLE,
        account_primary_where() /* where */ );

    ACCOUNT *account_parse(
        input,
        account_chart_account_boolean(),
        fetch_subclassification,
        fetch_element );
}

/* Usage */
ACCOUNT *account_parse(
    char *input,
    boolean account_chart_account_boolean,
    boolean fetch_subclassification,
    boolean fetch_element );

/* Process */
if ( fetch_subclassification )
{
    account->subclassification =
        subclassification_fetch(
            account->subclassification_name,
            fetch_element );
}

/* Usage */
LIST *account_list(
    char *where,
    boolean fetch_subclassification,
    boolean fetch_element );

/* Process */
boolean account_chart_account_boolean(
    ACCOUNT_TABLE,
    ACCOUNT_CHART_ACCOUNT_NUMBER );

char *account_select_string(
    ACCOUNT_SELECT,
    ACCOUNT_CHART_ACCOUNT_NUMBER,
    account_chart_account_boolean() );

char *account_system_string(
    account_select_string(),
    ACCOUNT_TABLE,
    where );

LIST *account_system_list(
    account_system_string(),
    account_chart_account_boolean(),
    fetch_subclassification,
    fetch_element );

/* Usage */
LIST *account_system_list(
    char *account_system_string(),
    boolean account_chart_account_boolean,
    boolean fetch_subclassification,
    boolean fetch_element );

/* Process */
FILE *account_pipe( char *account_system_string() );

for input[] in string_input( account_pipe() )
{
    list_set(
        list,
        account_parse(
            input,
            account_chart_account_boolean,
            fetch_subclassification,
            fetch_element ) );
}

pclose( account_pipe() );

/* Usage */
LIST *account_balance_sort_list(
    LIST *account_statement_list() );

/* Process */
int account_balance_compare_function(
    ACCOUNT *from_list_account,
    ACCOUNT *compare_account );

ACCOUNT_JOURNAL *from_list_latest_journal =
    from_list_account->account_journal_latest;

ACCOUNT_JOURNAL *compare_latest_journal =
    compare_account->account_journal_latest;

if ( !from_list_latest_journal
||   !from_list_latest_journal->balance )
{
    return 1;
}

if ( !compare_latest_journal
||   !compare_latest_journal->balance )
{
    return 1;
}

if (from_list_latest_journal->balance <=
    compare_latest_journal->balance )
{
    return 1;
}
else
{
    return -1;
}
```

ACCOUNT (3)

```
/* Usage */
void account_list_delta_prior_percent_set(
    LIST *prior_account_list in/out */,
    LIST *current_account_list );

/* Process */
for ACCOUNT *current_account in
    current_account_list
{
    ACCOUNT *prior_account =
       ACCOUNT *account_seek(
          current_account->account_name,
          prior_account_list /* account_list */ );

    prior_account->delta_prior_percent =
       float_delta_prior_percent(
          prior_account->
             account_journal_latest->
             balance,
          current_account->
             account_journal_latest->
             balance );
}

/* Usage */
void account_percent_of_asset_set(
    LIST *subclassification_account_statement_list,
    double element_asset_sum );

/* Process */
for ACCOUNT *account in
    subclassifiication_account_statement_list
{
    int account->percent_of_asset =
       float_percent_of_total(
          account->
             account_journal_latest->
             balance,
          element_asset_sum );
}

/* Usage */
void account_percent_of_income_set(
    LIST *subclassification_account_statement_list,
    double element_income );

/* Process */
for ACCOUNT *account in
    subclassification_account_statement_list
{
    int account->percent_of_income =
       float_percent_of_total(
          account->
             account_journal_latest->
             balance,
          element_income );
}

/* Usage */
boolean account_accumulate_debit(
    char *account_name );

/* Process */
ACCOUNT *account_fetch(
    account_name,
    1 /* fetch_subclassification */,
    1 /* fetch_element */ );

return
account->
    subclassification->
    element->
    accumulate_debit;
```

ACCOUNT (4)

```
/* Usage */
double account_list_debit_sum(
   LIST *account_list,
   boolean element_accumulate_debit );

/* Process */
for ACCOUNT *account in account_list
{
   if (   !element_accumulate_debit
   &&  account_journal_latest->balance < 0.0 )
   {
      debit_sum -=
         account_journal_latest->balance;
   }
   else
   if (   element_accumulate_debit
   &&  account_journal_latest->balance > 0.0 )
   {
      debit_sum +=
         account_journal_latest->balance;
   }
}

/* Usage */
double account_list_credit_sum(
   LIST *account_list,
   boolean element_accumulate_debit );

/* Process */
for ACCOUNT *account in account_list
{
   if (   element_accumulate_debit
   &&  account_journal_latest->balance < 0.0 )
   {
      credit_sum -=
         account_journal_latest->balance;
   }
   else
   if (   !element_accumulate_debit
   &&  account_journal_latest->balance > 0.0 )
   {
      credit_sum +=
         account_journal_latest->balance;
   }
}

/* Usage */
double account_balance_sum(
   LIST *account_list );

/* Process */
for ACCOUNT *account in account_list
{
   balance_sum += account_journal_latest->balance;
}

/* Usage */
char *account_hard_coded_account_name(
   char *account_key,
   boolean warning_only,
   const char *calling_function_name );

/* Process */
if ( !account_list )
{
   static LIST *account_list();
}

ACCOUNT *account_key_seek(
   account_key,
   account_list() );

/* Usage */
ACCOUNT *account_parse(
   char *input,
   boolean account_chart_account_boolean,
   boolean fetch_subclassification,
   boolean fetch_element );

/* Process */
ACCOUNT *account_new( account_name );

if ( fetch_subclassification )
{
   subclassification =
      subclassification_fetch(
         account->subclassification_name,
         fetch_element );
}

/* Usage */
ACCOUNT *account_new(
   char *account_name );

/* Process */
ACCOUNT *account_calloc(
   void );

/* Usage */
double account_list_sum(
   LIST *account_statement_list );

/* Process */

/* Usage */
ACCOUNT *account_element_list_seek(
   char *account_name,
   LIST *element_statement_list );

/* Process */
for ELEMENT *element in element_statement_list
{
   ACCOUNT *account_subclassification_account_seek(
      account_name,
      element->subclassification_statement_list );
}

/* Usage */
ACCOUNT *account_subclassification_list_seek(
   char *account_name,
   LIST *subclassification_statement_list );

/* Process */
for SUBCLASSIFICATION *subclassification in
   subclassification_statement_list
{
   ACCOUNT *account_seek(
      account_name,
      subclassification->account_statement_list );
}
```

ACCOUNT (5)

```
/* Usage */
ACCOUNT *account_key_seek(
    char *account_key,
    static LIST *account_list() );

/* Process */
for ACCOUNT *account in account_list
{
    boolean piece_boolean(
       account_key /* search_string */,
       account->hard_coded_account_key
          /* delimited_string */,
       '|' /* delimiter );

    if ( piece_boolean() ) return account;
}
return NULL;

/* Usage */
void account_transaction_count_set(
    LIST *account_statement_list,
    char *transaction_begin_date_string(),
    char *end_date_time_string );

/* Process */
for ACCOUNT *account in account_statement_list
{
    if ( account->journal_latest->balance )
    {
       account->transaction_count =
          journal_transaction_count(
             JOURNAL_TABLE,
             account->account_name,
             transaction_begin_date_string,
             end_date_time_string );
    }
}

/* Usage */
void account_list_action_string_set(
    LIST *account_statement_list,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *transaction_begin_date_string(),
    char *end_date_time_string );

/* Process */
for ACCOUNT *account in account_statement_list
{
    if ( account->transaction_count )
    {
       account->action_string =
          char *account_action_string(
             const char *JOURNAL_TABLE,
             const char *
                DICTIONARY_SEPARATE_DRILLTHRU_PREFIX,
             const char *DRILLTHRU_SKIPPED_KEY,
             const char *PROMPT_LOOKUP_FROM_PREFIX,
             const char *PROMPT_LOOKUP_TO_PREFIX,
             char *application_name,
             char *session_key,
             char *login_name,
             char *role_name,
             char *transaction_begin_date_string,
             char *end_date_time_string,
             char *account->account_name );
    }
}

/* Usage */
char *account_closing_entry_string(
    const char *ACCOUNT_CLOSING_KEY,
    const char *ACCOUNT_EQUITY_KEY,
    const char *calling_function_name );

/* Process */
char *account_hard_coded_account_name(
    ACCOUNT_CLOSING_KEY,
    1 /* warning_only */,
    calling_function_name );

if ( account_hard_coded_account_name() ) return this;
char *account_hard_coded_account_name(
    ACCOUNT_EQUITY_KEY,
    0 /* not warning_only */,
    calling_function_name );

/* Usage */
char *account_key_account_name(
    const char *account_key,
    const char *calling_function_name );

/* Process */
char *account_hard_coded_account_name(
    account_key,
    0 /* not warning_only */,
    calling_function_name );

/* Usage */
boolean account_chart_account_boolean(
    const char *ACCOUNT_TABLE,
    const char *ACCOUNT_CHART_ACCOUNT_NUMBER );

/* Process */
boolean folder_attribute_exists(
    ACCOUNT_TABLE,
    ACCOUNT_CHART_ACCOUNT_NUMBER );

/* Usage */
char *account_select_string(
    const char *ACCOUNT_SELECT,
    const char *ACCOUNT_CHART_ACCOUNT_NUMBER,
    boolean account_chart_account_boolean() );

/* Process */

/* Usage */
char *account_system_string(
    char *account_select_string(),
    const char *ACCOUNT_TABLE,
    char *where );

/* Process */
```

ACCOUNT (6)

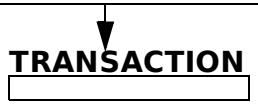
```
/* Usage */
ACCOUNT *account_drawing(
    const char *ACCOUNT_DRAWING_KEY,
    const char *calling_function_name );

/* Process */
ACCOUNT *account_key_fetch(
    ACCOUNT_DRAWING_KEY,
    calling_function_name );

/* Usage */
ACCOUNT *account_depreciation(
    const char *ACCOUNT_DEPRECIATION_KEY,
    const char *calling_function_name );

/* Process */
ACCOUNT *account_key_fetch(
    ACCOUNT_DEPRECIATION_KEY,
    calling_function_name );

/* Usage */
ACCOUNT *account_accumulated_depreciation(
    const char *ACCOUNT_ACCUMULATED_KEY,
    const char *calling_function_name );

/* Process */
ACCOUNT *account_key_fetch(
    ACCOUNT_ACCUMULATED_KEY,
    calling_function_name );

/* Usage */
ACCOUNT *account_sales_tax_payable(
    const char *ACCOUNT_SALES_TAX_PAYABLE_KEY,
    const char *calling_function_name );

/* Process */
ACCOUNT *account_key_fetch(
    ACCOUNT_SALES_TAX_PAYABLE_KEY,
    calling_function_name );

/* Usage */
ACCOUNT *account_revenue(
    const char *ACCOUNT_REVENUE_KEY,
    const char *calling_function_name );

/* Process */
ACCOUNT *account_key_fetch(
    ACCOUNT_REVENUE_KEY,
    calling_function_name );

/* Usage */
ACCOUNT *account_receivable(
    const char *ACCOUNT_RECEIVABLE_KEY,
    const char *calling_function_name );

/* Process */
ACCOUNT *account_key_fetch(
    ACCOUNT_RECEIVABLE_KEY,
    calling_function_name );

/* Usage */
ACCOUNT *account_payable(
    const char *ACCOUNT_PAYABLE_KEY,
    const char *calling_function_name );

/* Process */
ACCOUNT *account_key_fetch(
    ACCOUNT_PAYABLE_KEY,
    calling_function_name );

/* Usage */
ACCOUNT *account_uncleared_checks(
    const char *ACCOUNT_UNCLEARED_CHECKS_KEY,
    const char *calling_function_name );

/* Process */
ACCOUNT *account_key_fetch(
    ACCOUNT_UNCLEARED_CHECKS_KEY,
    calling_function_name );

/* Usage */
ACCOUNT *account_shipping_revenue(
    const char *ACCOUNT_SHIPPING_REVENUE_KEY,
    const char *calling_function_name );

/* Process */
ACCOUNT *account_key_fetch(
    ACCOUNT_SHIPPING_REVENUE_KEY,
    calling_function_name );

/* Usage */
ACCOUNT *account_gain(
    const char *ACCOUNT_GAIN_KEY,
    const char *calling_function_name );

/* Process */

/* Usage */
ACCOUNT *account_loss(
    const char *ACCOUNT_LOSS_KEY,
    const char *calling_function_name );

/* Process */

/* Usage */
ACCOUNT *account_fees_expense(
    const char *ACCOUNT_FEES_EXPENSE_KEY,
    const char *calling_function_name );

/* Process */

/* Usage */
ACCOUNT *account_sales_tax_expense(
    const char *ACCOUNT_SALES_TAX_EXPENSE_KEY,
    const char *calling_function_name );

/* Process */

/* Usage */
ACCOUNT *account_equity(
    const char *ACCOUNT_EQUITY_KEY,
    const char *calling_function_name );

/* Process */

/* Usage */
ACCOUNT *account_credit_card_passthru(
    const char *ACCOUNT_PASSTHRU_KEY,
    const char *calling_function_name );

/* Process */

/* Usage */
ACCOUNT *account_cost_of_goods_sold(
    const char *ACCOUNT_CGS_KEY,
    const char *calling_function_name );

/* Process */

/* Usage */
ACCOUNT *account_inventory(
    const char *ACCOUNT_INVENTORY_KEY,
    const char *calling_function_name );

/* Process */
```

ACCOUNT (7)

```
/* Usage */
char *account_drawing_string(
    const char *ACCOUNT_DRAWING_KEY );

/* Process */
char *account_hard_coded_account_name(
    ACCOUNT_DRAWING_KEY,
    1 /* warning_only */,
    __FUNCTION__ );

/* Usage */
char *account_equity_string(
    const char *ACCOUNT_EQUITY_KEY,
    const char *calling_function_name );

/* Process */
char *account_hard_coded_account_name(
    ACCOUNT_EQUITY_KEY,
    0 /* not warning_only */,
    calling_function_name );

/* Usage */
char *account_uncleared_checks_string(
    const char *ACCOUNT_UNCLEARED_CHECKS_KEY,
    const char *calling_function_name );

/* Process */
char *account_hard_coded_account_name(
    ACCOUNT_UNCLEARED_CLEARED_KEY,
    0 /* not warning_only */,
    calling_function_name );

/* Usage */
char *account_depreciation_string(
    const char *ACCOUNT_DEPRECIATION_KEY,
    const char *calling_function_name );

/* Process */
char *account_hard_coded_account_name(
    ACCOUNT_DEPRECIATION_KEY,
    0 /* not warning_only */,
    calling_function_name );

/* Usage */
char *account_accumulated_depreciation_string(
    const char *
      ACCOUNT_ACCUMULATED_DEPRECIATION_KEY,
    const char *calling_function_name );

/* Process */
char *account_hard_coded_account_name(
    ACCOUNT_ACCUMULATED_DEPRECIATION_KEY,
    0 /* not warning_only */,
    calling_function_name );

/* Usage */
char *account_payable_string(
    const char *ACCOUNT_PAYABLE_KEY,
    const char *calling_function_name );

/* Process */

/* Usage */
char *account_receivable_string(
    const char *ACCOUNT_RECEIVABLE_KEY,
    const char *calling_function_name );

/* Process */

/* Usage */
char *account_loss_string(
    const char *ACCOUNT_LOSS_KEY,
    const char *calling_function_name );

/* Process */

/* Usage */
char *account_name_format(
    char *account_name );

/* Process */

/* Usage */
char *account_name_display(
    char *account_name );

/* Process */

/* Usage */
ACCOUNT *account_seek(
    char *account_name,
    LIST *account_list );

/* Process */

/* Usage */
LIST *account_cash_name_list(
    const char *ACCOUNT_TABLE,
    const char *SUBCLASSIFICATION_CASH );

/* Process */

/* Usage */
LIST *account_current_liability_name_list(
    const char *ACCOUNT_TABLE,
    const char *SUBCLASSIFICATION_CURRENT_LIABILITY,
    const char *ACCOUNT_CREDIT_CARD_KEY,
    LIST *exclude_account_name_list );

/* Process */

/* Usage */
LIST *account_receivable_name_list(
    const char *ACCOUNT_TABLE,
    const char *SUBCLASSIFICATION_RECEIVABLE );

/* Process */

/* Usage */
ACCOUNT *account_getset(
    LIST *account_list,
    ACCOUNT *account );

/* Process */

/* Usage */
boolean account_name_changed(
    char *preupdate_account_name );

/* Process */
```

ACCOUNT_JOURNAL

```
/* Usage */
ACCOUNT_JOURNAL *account_journal_latest(
    char *account_name,
    char *end_date_time_string,
    boolean fetch_transaction );

/* Process */
ACCOUNT_JOURNAL *account_journal_calloc(
    void );

JOURNAL *journal_latest(
    JOURNAL_TABLE,
    account_name,
    end_date_time_string,
    fetch_transaction,
    0 /* not zero_balance_boolean */ );

if ( !journal_latest() ) return null;

boolean float_virtually_same(
    journal_latest()->balance,
    0.0 );

if ( float_virtually_same() ) return null;

/* Attributes */
char *full_name;
char *street_address;
char *transaction_date_time;
char *account_name;
double previous_balance;
double debit_amount;
double credit_amount;
double balance;
TRANSACTION *transaction;
```



JOURNAL (1)

```
/* Usage */
LIST *journal_system_list(
    boolean fetch_account,
    boolean fetch_subclassification,
    boolean fetch_element,
    boolean fetch_transaction );

/* Process */
char *journal_system_string(
     JOURNAL_SELECT,
     JOURNAL_TABLE,
     (char *)0 /* where */ );

FILE *journal_input_pipe(
    journal_system_string() );

for input in string_input( journal_input_pipe() )
{
    list_set(
       system_list,
       journal_parse(
          input,
          fetch_account,
          fetch_subclassification,
          fetch_element,
          fetch_transaction ) );
}

void pclose( journal_input_pipe() );
```

```
/* Usage */
JOURNAL *journal_parse(
    char *input,
    boolean fetch_account,
    boolean fetch_subclassification,
    boolean fetch_element,
    boolean fetch_transaction );

/* Process */
JOURNAL *journal = journal_new(
    full_name,
    street_address,
    transaction_date_time,
    account_name );

if ( fetch_transaction )
{
    TRANSACTION *transaction =
       transaction_fetch(
          journal->full_name,
          journal->street_address,
          journal->transaction_date_time,
          0 /* not fetch_journal_list */ );
}

if ( fetch_account )
{
    ACCOUNT *account =
       account_fetch(
          journal->account_name,
          fetch_subclassification,
          fetch_element );
}

/* Usage */
JOURNAL *journal_new(
    char *full_name,
    char *street_address,
    char *transaction_date_time,
    char *account_name );

/* Process */
JOURNAL *journal_calloc(
    void );

/* Attributes */
char *full_name;
char *street_address;
char *transaction_date_time;
char *account_name;

/* Set externally */
double previous_balance;
double debit_amount;
double credit_amount;
double balance;
TRANSACTION *transaction;
ACCOUNT *account;
```

TRANSACTION

ACCOUNT

```
SUBCLASSIFICATION *subclassification;
```

JOURNAL (2)

```
/* Usage */
JOURNAL *journal_prior(
    char *transaction_date_time,
    char *account_name,
    boolean fetch_account,
    boolean fetch_subclassification,
    boolean fetch_element );

/* Process */
char *journal_max_prior_where(
    char *transaction_date_time,
    char *account_name );

char *journal_max_prior_transaction_date_time(
    char *journal_max_prior_where(),
    char *JOURNAL_TABLE );

if ( journal_max_prior_transaction_date_time() )
       return NULL;

JOURNAL *journal =
    journal_account_fetch(
       journal_max_prior_transaction_date_time(),
       account_name,
       fetch_account,
       fetch_subclassification,
       fetch_element,
       0 /* not fetch_transaction */ );

/* Usage */
JOURNAL *journal_latest(
    const char *JOURNAL_TABLE,
    char *account_name,
    char *end_date_time_string,
    boolean fetch_transaction_boolean,
    boolean zero_balance_boolean );

/* Process */
char *journal_less_equal_where(
    char *end_date_time_string,
    char *account_name );

char *journal_maximum_transaction_date_time(
    const char *JOURNAL_TABLE,
    char *journal_less_equal_where() );

JOURNAL *journal_account_fetch(
    journal_max_transaction_date_time()
       /* transaction_date_time */,
    account_name,
    0 /* not fetch_account */,
    0 /* not fetch_subclassification */,
    0 /* not fetch_element */,
    fetch_transaction_boolean );

if (    !zero_balance_boolean
&&   journal_account_fetch()
&&   !journal_account_fetch()->balance )
{
    return null;
}

/* Usage */
LIST *journal_year_list(
    int tax_year,
    char *tax_form_line_account->account_name,
    boolean fetch_transaction );

/* Process */
char *journal_year_where(
    int tax_year,
    char *account_name,
    const char *TRANSACTION_DATE_PRECLOSE_TIME );

char *journal_system_string(
    JOURNAL_SELECT,
    JOURNAL_TABLE,
    journal_year_where(),
    0 /* not fetch_account */,
    0 /* not fetch_subclassification */,
    0 /* not fetch_element */,
    fetch_transaction );

LIST *journal_system_list(
    journal_system_string() );

/* Usage */
JOURNAL *journal_account_fetch(
    char *transaction_date_time,
    char *account_name,
    boolean fetch_account,
    boolean fetch_subclassification,
    boolean fetch_element,
    boolean fetch_transaction );

/* Process */
char *journal_transaction_account_where(
    char *transaction_date_time,
    char *account_name );

JOURNAL *journal_parse(
    string_pipe(
       journal_system_string(
          JOURNAL_SELECT,
          JOURNAL_TABLE,
          journal_transaction_account_where() ),
    fetch_account,
    fetch_subclassification,
    fetch_element,
    fetch_transaction );

/* Usage */
void journal_account_list_propagate(
    char *transaction_date_time,
    LIST *journal_extract_account_list() );

/* Process */
for ACCOUNT *account in journal_extract_account_list
{
    JOURNAL_PROPAGATE *
       journal_propagate_new(
          transaction_date_time,
          account->account_name );

    if ( journal_propagate_new() )
    {
       journal_list_update(
          journal_propagate_new()->
             update_statement_list );
    }
}
```

JOURNAL (3)

```
/* Usage */
void journal_account_list_getset(
    LIST *account_list /* in/out */,
    LIST *journal_list );

/* Process */
for JOURNAL *journal in journal_list
{
    (void)account_getset(
        account_list /* in/out */,
        journal->account );
}

/* Usage */
void journal_list_update(
    LIST *update_statement_list );

/* Process */
FILE *appaserver_output_pipe( “sql.e” );

for char *update_statement in update_statement_list
{
    fprintf(
        appaserver_output_pipe(),
        “%s\n”,
        update_statement );

}

void pclose( appaserver_output_pipe() );
```

```
/* Usage */
void journal_list_insert(
    char *full_name,
    char *street_address,
    char *transaction_date_time,
    LIST *journal_list );

/* Process */
FILE *journal_insert_pipe(
    char *JOURNAL_INSERT,
    char *JOURNAL_TABLE );

void journal_insert(
    FILE *journal_insert_pipe(),
    char *full_name,
    char *street_address,
    char *transaction_date_time,
    char *account_name,
    double debit_amount,
    double credit_amount );

void pclose( journal_insert_pipe() );

/* Usage */
void journal_list_transaction_insert(
    FILE *journal_insert_pipe(),
    char *full_name,
    char *street_address,
    char *transaction_date_time,
    LIST *journal_list );

/* Process */
for JOURNAL *journal in journal_list
{
    void journal_insert(
        journal_insert_pipe,
        full_name,
        street_address,
        transaction_date_time,
        journal->account_name,
        journal->debit_amount,
        journal->credit_amount );
}

/* Usage */
char *journal_system_string(
    const char *JOURNAL_SELECT,
    const char *JOURNAL_TABLE,
    char *where );

/* Process */
if ( !where ) where = “1 = 1”;

sprintf(
    system_string[],
    “select.sh \”%s\” %s \”%s\” transaction_date_time”,
    JOURNAL_SELECT,
    JOURNAL_TABLE,
    where );
```

JOURNAL (4)

```
/* Usage */
void journal_list_html_display(
    LIST *journal_list,
    char *transaction_date_time,
    char *transaction_memo,
    char *transaction_full_name );

/* Process */
void journal_list_pipe_display(
    FILE *output_pipe,
    char *transaction_date_time,
    char *transaction_memo,
    char *transaction_full_name,
    LIST *journal_list );

/* Usage */
void journal_list_sum_html_display(
    LIST *journal_list,
    char *transaction_date_time,
    char *transaction_memo,
    double debit_sum,
    double credit_sum );

/* Process */
void journal_list_pipe_display(
    output_pipe,
    transaction_date_time,
    transaction_memo,
    (char *)0 /* transaction_full_name */,
    journal_list );

/* Usage */
double journal_list_transaction_amount(
    LIST *journal_list );

/* Process */
static LIST *account_cash_name_list(
     ACCOUNT_TABLE,
     SUBCLASSIFICATION_CASH );

LIST *exclude_account_name_list = list_new();

list_set(exclude_account_name_list,
        ACCOUNT_UNCLEARED_CHECKS );

static LIST *account_current_liability_name_list(
     ACCOUNT_TABLE,
     SUBCLASSIFICATION_CURRENT_LIABILITY,
     ACCOUNT_CREDIT_CARD_KEY,
     exclude_account_name_list );

double cash_sum =
    journal_account_list_debit_sum(
        journal_list,
        account_cash_name_list() );

double current_liability_sum =
    journal_account_list_credit_sum(
        journal_list,
        account_current_liability_name_list() );

if ( cash_sum )
{
    return cash_sum;
}
else
if ( current_liability_sum )
{
    return current_liability_sum;
}
else
{
    return journal_debit_sum( journal_list );
}

/* Usage */
double journal_account_list_debit_sum(
    LIST *journal_list,
    LIST *account_name_list );

/* Process */
for JOURNAL *journal in journal_list
{
    if ( list_string_exists(
        journal->account_name,
        account_name_list ) )
    {
        if ( journal->debit_amount )
            sum += journal->debit_amount;
        else
            sum -= journal->credit_amount;
    }
}

/* Usage */
double journal_account_list_credit_sum(
    LIST *journal_list,
    LIST *account_name_list );

/* Process */
for JOURNAL *journal in journal_list
{
    if ( list_string_exists(
        journal->account_name,
        account_name_list ) )
    {
        if ( journal->credit_amount )
            sum += journal->credit_amount;
        else
            sum -= journal->debit_amount;
    }
}

/* Usage */
double journal_first_account_balance(
    char *account_name );

/* Process */
char *journal_minimum_transaction_date_time(
    account_name );

if ( !journal_minimum_transaction_date_time() ) exit( 1 );

JOURNAL *journal_account_fetch(
     journal_minimum_transaction_date_time(),
     account_name,
     0 /* not fetch_account */,
     0 /* not fetch_subclassification */,
     0 /* not fetch_element */,
     0 /* not fetch_transaction */ );

return journal_account_fetch()->balance;
```

JOURNAL (5)

```
/* Usage */
LIST *journal_tax_form_list(
    char *tax_form_fiscal_begin_date(),
    char *tax_form_fiscal_end_date(),
    const char *TRANSACTION_DATE_PRECLOSE_TIME,
    char *account_name );

/* Process */
char *journal_tax_form_where(
    char *tax_form_fiscal_begin_date(),
    char *tax_form_fiscal_end_date(),
    const char *TRANSACTION_DATE_PRECLOSE_TIME,
    char *account_name );

char *journal_system_string(
    JOURNAL_SELECT,
    JOURNAL_TABLE,
    journal_tax_form_where() );

LIST *journal_system_list(
    journal_system_string(),
    0 /* not fetch_account */,
    0 /* not fetch_subclassification */,
    0 /* not fetch_element */,
    0 /* not fetch_transaction */ );
```

```
/* Usage */
double journal_debit_sum( LIST *journal_list );

/* Process */
for JOURNAL *Journal in journal_list
{
   if ( journal->debit_amount )
      sum += journal->debit_amount;
}

/* Usage */
double journal_credit_sum( LIST *journal_list );

/* Process */
for JOURNAL *journal in journal_list
{
    if ( journal->credit_amount )
       sum += journal->credit_amount;
}

/* Usage */
double journal_debit_credit_sum_difference(
   boolean element_accumulate_debit,
   LIST *journal_list );

/* Process */
```

```
double journal_debit_sum( journal_list );

double journal_credit_sum( journal_list );

if ( element_accumulate_debit )
{
   journal_debit_sum() - journal_credit_sum();
}
else
{
   journal_credit_sum() - journal_debit_sum();
}

/* Usage */
double journal_credit_debit_difference_sum(
   LIST *journal_list );

/* Process */

/* Usage */
double journal_debit_credit_difference_sum(
   LIST *journal_list );

/* Process */
```

JOURNAL (6)

```
/* Usage */
JOURNAL *journal_account_new(
    double journal_amount,
    ACCOUNT *debit_account,
    ACCOUNT *credit_account );

/* Process */
JOURNAL *journal_calloc();

if ( debit_account )
{
    journal_calloc()->debit_amount = journal_amount;
    journal_calloc()->account = debit_account;
}
else
{
    journal_calloc()->credit_amount = journal_amount;
    journal_calloc()->account = credit_account;
}

/* Usage */
JOURNAL *journal_debit_new(
    char *debit_account_name,
    double debit_amount );

/* Process */
JOURNAL *journal_calloc();

journal->account =
    ACCOUNT *account_fetch(
        debit_account_name,
        1 /* fetch_subclassification */,
        1 /* fetch_element */ );

/* Usage */
JOURNAL *journal_credit_new(
    char *credit_account_name,
    double credit_amount );

/* Process */
JOURNAL *journal_calloc();

journal->account =
    ACCOUNT *account_fetch(
        credit_account_name,
        1 /* fetch_subclassification */,
        1 /* fetch_element */ );

/* Usage */
LIST *journal_binary_list(
    char *full_name,
    char *street_address,
    char *transaction_date_time,
    double float_abs( transaction_amount ),
    ACCOUNT *debit_account,
    ACCOUNT *credit_account );

/* Process */

/* Usage */
char *journal_system_string(
    char *JOURNAL_SELECT,
    char *JOURNAL_TABLE,
    char *where );

/* Process */

/* Usage */
JOURNAL *journal_seek(
    char *transaction_date_time,
    char *account_name,
    char *journal_system_list() );

/* Process */

/* Usage */
LIST *journal_entity_list(
    const char *JOURNAL_SELECT,
    const char *JOURNAL_TABLE,
    char *full_name,
    char *street_address,
    char *account_name );j

/* Process */
char *journal_entity_where(
    char *full_name,
    char *street_address,
    char *account_name );

char *journal_system_string(
    JOURNAL_SELECT,
    JOURNAL_TABLE,
    journal_entity_where() );

LIST *journal_system_list(
    journal_system_string(),
    0 /* not fetch_account */,
    0 /* not fetch_subclassification */,
    0 /* not fetch_element */,
    0 /* not fetch_transaction */ );

/* Usage */
LIST *journal_transaction_list(
    const char *JOURNAL_SELECT,
    const char *JOURNAL_TABLE,
    char *full_name,
    char *street_address,
    char *transaction_date_time );

/* Process */
char *transaction_primary_where(
    char *full_name,
    char *street_address,
    char *transaction_date_time );

char *journal_system_string(
    JOURNAL_SELECT,
    JOURNAL_TABLE,
    transaction_primary_where() );

LIST *journal_system_list(
    journal_system_string(),
    1 /* fetch_account */,
    1 /* fetch_subclassification */,
    1 /* fetch_element */,
    0 /* not fetch_transaction */ );

/* Usage */
char *journal_primary_where(
    char *full_name,
    char *street_address,
    char *transaction_date_time,
    char *account_name );

/* Processs */
```

JOURNAL (7)

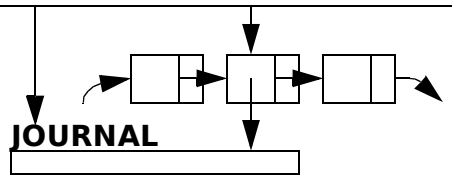
```
/* Usage */
boolean journal_list_match_boolean(
    LIST *journal1_list,
    LIST *journal2_list );

/* Process */
if (    list_length( journal1_list ) !=
        list_length( journal2_list ) )
{
    return 0;
}

for JOURNAL *journal1 in journal1_list
{
    boolean journal_exists_boolean(
        journal2_list /* journal_list */,
        journal1 /* match_journal */ );

    if ( !journal_exists_boolean() ) return 0;
}

return 1;

/* Usage */
boolean journal_exists_boolean(
    LIST *journal_list,
    JOURNAL *match_journal );

/* Process */
for JOURNAL *journal in journal_list
{
    boolean journal_match_boolean(
        match_journal /* journal1 */,
        journal /* journal2 */ );

    if ( journal_match_boolean() ) return 1;
}

return 0;

/* Usage */
boolean journal_match_boolean(
    JOURNAL *journal1,
    JOURNAL *journal2 );

/* Process */
if ( strcmp(
        journal1->account->account_name,
        journal2->account->account_name ) != 0 )
{
    return 0;
}

if ( journal1->debit_amount )
{
    boolean float_virtually_same(
        journal1->debit_amount,
        journal2->debit_amount );

    if ( !float_virtually_same() ) return 0;
}
else
if ( journal1->credit_amount )
{
    boolean float_virtually_same(
        journal1->credit_amount,
        journal2->credit_amount );

    if ( !float_virtually_same() ) return 0;
}

return 1;
```

JOURNAL (8)

```
/* Public */
LIST *journal_account_distinct_entity_list(
    char *JOURNAL_TABLE,
    LIST *account_name_list );

LIST *journal_extract_account_list(
    LIST *journal_list );

char *journal_delete_system_string(
    char *JOURNAL_TABLE,
    char *transaction_primary_where()
       /* where */ );

int journal_transaction_count(
    char *JOURNAL_TABLE,
    char *account->account_name,
    char *transaction_begin_date_string,
    char *transaction_date_time_closing );

LIST *journal_date_time_account_name_list(
    char *JOURNAL_TABLE,
    char *transaction_date_time );

char *journal_minimum_transaction_date_time(
    char *account_name );

double journal_signed_balance(
    double balance,
    boolean element_accumulate_debit );

LIST *journal_account_name_list(
    char *full_name,
    char *street_address,
    char *transaction_date_time );

double journal_amount(
    double debit_amount,
    double credit_amount,
    boolean element_accumulate_debit );
```

JOURNAL_PROPAGATE (1)

```
/* Usage */
JOURNAL_PROPAGATE *journal_propagate_new(
    char *transaction_date_time,
    char *account_name );

/* Process */
JOURNAL_PROPAGATE *journal_propagate_calloc(
    void );

char *string_atoi( transaction_date_time );

/* If doing period of record */
if ( !string_atoi() )
{
    transaction_date_time =
        journal_minimum_transaction_date_time(
            account_name );
}

if ( !transaction_date_time ) return NULL;

JOURNAL *journal_prior(
    transaction_date_time,
    account_name,
    1 /* fetch_subclassification */,
    1 /* fetch_element */,
    1 /* fetch_account */ );

char *journal_propagate_prior_transaction_date_time(
    journal_prior() );

double journal_propagate_prior_previous_balance(
    journal_prior() );

LIST *journal_propagate_journal_list(
    account_name,
    journal_propagate_prior_transaction_date_time(),
    journal_propagate_prior_previous_balance() );

if ( list_length( journal_propagate_journal_list() )
{
    boolean journal_propagate_accumulate_debit(
        account_name,
        journal_prior() );

    void journal_propagate_balance_set(
        journal_propagate_journal_list() /* in/out */,
        journal_propagate_accumulate_debit() );

    LIST *journal_propagate_update_statement_list(
        JOURNAL_TABLE,
        journal_propagate_journal_list() );
}

/* Attributes */
JOURNAL *journal_prior;
char *prior_transaction_date_time;
double prior_previous_balance;
boolean accumulate_debit;
LIST *journal_list;
LIST *update_statement_list;
```



JOURNAL_PROPAGATE (2)

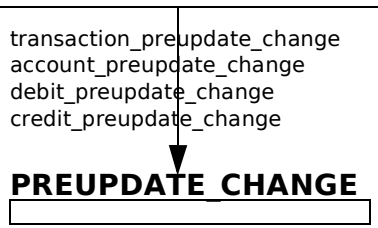
```
/* Usage */
char *journal_propagate_prior_transaction_date_time(
    JOURNAL *journal_prior() );

/* Process */
if ( journal_prior )
    return journal_prior->transaction_date_time;
else
    return (char *)0;

/* Usage */
double journal_propagate_prior_previous_balance(
    JOURNAL *journal_prior() );

/* Process */
if ( journal_prior )
    return journal_prior->previous_balance;
else
    return 0.0;

/* Usage */
boolean journal_propagate_accumulate_debit(
    char *account_name,
    JOURNAL *journal_prior() );

/* Process */
if ( journal_prior )
{
    boolean accumulate_debit =
        boolean journal_prior->
            account->
            subclassification->
            element->
            accumulate_debit;
}
else
{
    boolean accumulate_debit =
        boolean account_accumulate_debit(
            account_name );
}


/* Usage */
LIST *journal_propagate_journal_list(
    char *account_name,
    char *journal_propagate_prior_transaction_date_time,
    double journal_propagate_prior_previous_balance );

/* Process */
char *journal_propagate_greater_equal_where(
    account_name,
    journal_propagate_prior_transaction_date_time );

LIST *journal_list =
    journal_system_list(
        journal_system_string(
            JOURNAL_SELECT,
            JOURNAL_TABLE,
            journal_propagate_greater_equal_where() ),
        0 /* not fetch_account */,
        0 /* not fetch_subclassification */,
        0 /* not fetch_element */,
        0 /* not fetch_transaction */ );

int list_length( journal_list );

if ( list_length() )
{
    JOURNAL *first_journal = list_first( journal_list );

    first_journal->previous_balance =
        prior_previous_balance;
}

/* Usage */
void journal_propagate_balance_set(
    LIST *journal_list /* in/out */,
    boolean accumulate_debit_boolean );

/* Process */
JOURNAL *prior_journal = {0};

for JOURNAL *journal in journal_list
{
    if ( prior_journal )
    {
        journal->previous_balance =
            prior_journal->balance;
    }

    journal->balance =
        journal_propagate_balance(
            accumulate_debit,
            journal->previous_balance,
            journal->debit_amount,
            journal->credit_amount );

    prior_journal = journal;
}

/* Usage */
double journal_propagate_balance(
    boolean accumulate_debit,
    double previous_balance,
    double debit_amount,
    double credit_amount );

/* Process */
if ( accumulate_debit )
{
    if ( debit_amount )
    {
        double balance =
            previous_balance +
            debit_amount;
    }
    else
    {
        double balance =
            previous_balance -
            credit_amount;
    }
}
else
{
    if ( credit_amount )
    {
        double balance =
            previous_balance +
            credit_amount;
    }
    else
    {
        double balance =
            previous_balance -
            debit_amount;
    }
}
```

JOURNAL_PROPAGATE (3)

```
/* Usage */
void journal_propagate_previous_balance_set(
    LIST *journal_list /* in/out */,
    double end_balance );

/* Process */
JOURNAL *prior_journal = {0};

for JOURNAL *journal in journal_list descending
{
    if ( !prior_journal )
    {
       journal->balance = end_balance;
    }
    else
    {
       journal->balance =
          prior_journal->
             previous_balance;
    }

    journal->previous_balance =
       journal_propagate_previous_balance(
          journal->debit_amount,
          journal->credit_amount,
          journal->balance );

    prior_journal = journal;
}

/* Usage */
double journal_propagate_previous_balance(
    double debit_amount,
    double credit_amount,
    double balance );

/* Process */
if ( debit_amount )
{
    double previous_balance =
       balance -
       debit_amount;
}
else
{
    double previous_balance =
       balance +
       credit_amount;
}

/* Usage */
char *journal_propagate_greater_equal_where(
    char *account_name,
    char *prior_transaction_date_time );

/* Process */
if ( prior_transaction_date_time )
{
    sprintf(
       static where[],
       "account = '%s' and "
       "transaction_date_time >= '%s'",
       account_name,
       prior_transaction_date_time );
}
else
{
    sprintf(
       static where[],
       "account = '%s'",
       account_name );
}

/* Usage */
LIST *journal_propagate_update_statement_list(
    const char *JOURNAL_TABLE,
    LIST *journal_propagate_journal_list );

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

for JOURNAL *journal in journal_propagate_journal_list
{
    char *journal_propagate_update_statement(
       JOURNAL_TABLE,
       journal->full_name,
       journal->street_address,
       journal->transaction_date_time,
       journal->account_name,
       journal->previous_balance,
       journal->balance );

    list_set(
       update_statement_list,
       journal_propagate_update_statement() );
}

/* Usage */
char *journal_propagate_update_statement(
    const char *JOURNAL_TABLE,
    char *full_name,
    char *street_address,
    char *transaction_date_time,
    char *account_name,
    double previous_balance,
    double balance );

/* Process */
char *journal_primary_where(
    full_name,
    street_address,
    transaction_date_time,
    account_name );

snprintf(
    update_statement[],
    sizeof ( update_statement ),
    "update %s "
    "set previous_balance = %.2lf, "
    "balance = %.2lf "
    "where %s;",
    JOURNAL_TABLE,
    previous_balance,
    balance,
    journal_primary_where() );

free( journal_primary_where() );
```

JOURNAL_TRIGGER (1)

```
/* Driver */
if ( strcmp(
       state,
       APPASERVER_PREDELETE_STATE ) == 0 )
{
   exit( 0 );
}

if ( strcmp(
       state,
       APPASERVER_DELETE_STATE ) == 0 )
{
   void journal_trigger_delete(
      full_name,
      street_address,
      transaction_date_time,
      account_name );

   exit( 0 );
}

if ( strcmp(
       state,
       APPASERVER_INSERT_STATE ) == 0 )
{
   void journal_trigger_insert(
      full_name,
      street_address,
      transaction_date_time,
      account_name );

   exit( 0 );
}

/* Must be APPASERVER_UPDATE_STATE */
PREUPDATE_CHANGE *transaction_preupdate_change =
   preupdate_change_new(
      APPASERVER_INSERT_STATE,
      APPASERVER_PREDELETE_STATE,
      state,
      preupdate_transaction_date_time,
      transaction_date_time /* postupdate_datum */,
      "preupdate_transaction_date_time"
         /* preupdate_placeholder_name */ );

PREUPDATE_CHANGE *account_preupdate_change =
   preupdate_change_new(
      APPASERVER_INSERT_STATE,
      APPASERVER_PREDELETE_STATE,
      state,
      preupdate_account,
      account_name /* postupdate_datum */,
      "preupdate_account"
         /* preupdate_placeholder_name */ );

PREUPDATE_CHANGE *debit_preupdate_change =
   preupdate_change_new(
      APPASERVER_INSERT_STATE,
      APPASERVER_PREDELETE_STATE,
      state,
      preupdate_debit_amount,
      debit_amount /* postupdate_datum */,
      "preupdate_debit_amount"
         /* preupdate_placeholder_name */ );

PREUPDATE_CHANGE *credit_preupdate_change =
   preupdate_change_new(
      APPASERVER_INSERT_STATE,
      APPASERVER_PREDELETE_STATE,
      state,
      preupdate_credit_amount,
      credit_amount /* postupdate_datum */,
      "preupdate_credit_amount"
         /* preupdate_placeholder_name */ );

if (   transaction_preupdate_change->
          no_change_boolean
&&  account_preupdate_change->
          no_change_boolean
&&  debit_preupdate_change->
          no_change_boolean
&&  credit_preupdate_change->
          no_change_boolean )
{
   exit( 0 );
}

void journal_trigger_update(
   full_name,
   street_address,
   transaction_date_time,
   account_name,
   preupdate_transaction_date_time,
   preupdate_account_name,
   transaction_preupdate_change->
      no_change_boolean,
   account_preupdate_change->
      no_change_boolean );
```



JOURNAL_TRIGGER (2)

```
/* Usage */
void journal_trigger_delete(
    char *full_name,
    char *street_address,
    char *transaction_date_time,
    char *account_name );

/* Process */
JOURNAL_PROPAGATE *journal_propagate_new(
    transaction_date_time,
    account_name );

/* If still rows for this account */
if ( journal_propagate_new() )
{
    void journal_list_update(
       journal_propagate->
          update_statement_list );
}

void transaction_fetch_update(
    full_name,
    street_address,
    transaction_date_time );

/* Usage */
void journal_trigger_insert(
    char *full_name,
    char *street_address,
    char *transaction_date_time,
    char *account_name );

/* Process */
JOURNAL_PROPAGATE *journal_propagate_new(
    transaction_date_time,
    account_name );

if ( !journal_propagate_new() ) exit( 1 );

void journal_list_update(
    journal_propagate->
       update_statement_list );

void transaction_fetch_update(
    full_name,
    street_address,
    transaction_date_time );

/* Usage */
void journal_trigger_update(
    char *full_name,
    char *street_address,
    char *transaction_date_time,
    char *account_name,
    char *preupdate_transaction_date_time,
    char *preupdate_account,
    boolean transaction_no_change_boolean,
    boolean account_no_change_boolean );

/* Process */
char *journal_trigger_earlier_date_time(
    char *transaction_date_time,
    char *preupdate_transaction_date_time,
    boolean transaction_no_change_boolean );

if ( !account_no_change_boolean )
{
    JOURNAL_PROPAGATE *journal_propagate_new(
          journal_trigger_earlier_date_time(),
          preupdate_account );

    journal_list_update(
       journal_propagate_new()->
          update_statement_list );
}

JOURNAL_PROPAGATE *journal_propagate_new(
       journal_trigger_earlier_date_time(),
       account_name );

journal_list_update(
    journal_propagate_new()->
       update_statement_list );

void transaction_fetch_update(
    full_name,
    street_address,
    transaction_date_time );
```

TRANSACTION (1)

```
/* Usage */
LIST *transaction_list(
    char *where,
    boolean fetch_journal_list );

/* Process */
char *transaction_system_string(
    TRANSACTION_SELECT,
    TRANSACTION_TABLE,
    where );

FILE *transaction_input_pipe(
    char *transaction_system_string() );

for input[] in string_input( transaction_input_pipe() )
{
    list_set(
       list,
       transaction_parse(
          input,
          fetch_journal_list ) );
}

void pclose( transaction_input_pipe() );

/* Usage */
TRANSACTION *transaction_fetch(
    char *full_name,
    char *street_address,
    char *transaction_date_time,
    boolean fetch_journal_list );

/* Process */
char *transaction_primary_where(
    full_name,
    street_address,
    transaction_date_time );

char *transaction_system_string(
    TRANSACTION_SELECT,
    TRANSACTION_TABLE,
    transaction_primary_where()
       /* where */ );

TRANSACTION *transaction_parse(
    string_pipe_fetch(
       transaction_system_string() )
          /* input */,
    fetch_journal_list );

/* Usage */
TRANSACTION *transaction_parse(
    char *input,
    boolean fetch_journal_list );

/* Process */
TRANSACTION *transaction_new(
    strdup( full_name ),
    strdup( street_address ),
    strdup( transaction_date_time ) );

if ( fetch_journal_list )
{
    char *transaction_primary_where(
       full_name,
       street_address,
       transaction_date_time );

    LIST *journal_list =
       journal_system_list(
          journal_system_string(
             JOURNAL_SELECT,
             JOURNAL_TABLE,
             transaction_primary_where() ),
          1 /* fetch_account */,
          1 /* fetch_subclassification */,
          1 /* fetch_element */,
          0 /* not fetch_transaction */ );
}

/* Usage */
TRANSACTION *transaction_new(
    char *full_name,
    char *street_address,
    char *transaction_date_time );

/* Process */
TRANSACTION *transaction_calloc(
    void );

/* Usage
TRANSACTION *transaction_entity_new(
    ENTITY *entity,
    char *transaction_date_time,
    double transaction_amount,
    int check_number,
    char *memo,
    LIST *journal_list );

/* Process */
TRANSACTION *transaction_calloc();

/* Usage */
TRANSACTION *transaction_binary(
    char *full_name,
    char *street_address,
    char *transaction_date_time,
    double transaction_amount,
    char *memo,
    char *debit_account_name,
    char *credit_account_name );

/* Process */
TRANSACTION *transaction_new(
    full_name,
    street_address,
    transaction_date_time );

LIST *journal_list =
    journal_binary_list(
       full_name,
       street_address,
       transaction_date_time,
       float_abs( transaction_amount ),
       account_fetch(
          debit_account_name,
          1 /* fetch_subclassification */,
          1 /* fetch_element */ ),
       account_fetch(
          credit_account_name,
          1 /* fetch_subclassification */,
          1 /* fetch_element */ ) );
```

TRANSACTION (2)

```
/* Usage */
/* May reset transaction->transaction_date_time */
void transaction_list_insert(
    LIST *transaction_list,
    boolean insert_journal_list_boolean,
    boolean transaction_lock_boolean );

/* Process */
char transaction_lock_yn(
    boolean transaction_lock_boolean );

for TRANSACTION *transaction in transaction_list
{
    transaction->transaction_date_time =
        transaction_insert(
            transaction->full_name,
            transaction->street_address,
            transaction->transaction_date_time,
            transaction->transaction_amount,
            transaction->check_number,
            transaction->memo,
            transaction_lock_yn(),
            transaction->journal_list,
            insert_journal_list_boolean );
}

* Usage */
/* Returns inserted transaction_date_time */
char *transaction_insert(
    char *full_name,
    char *street_address,
    char *transaction_date_time,
    double transaction_amount,
    int check_number,
    char *memo,
    char transaction_lock_yn,
    LIST *journal_list,
    boolean insert_journal_list_boolean );

/* Process */
if ( transaction_amount <= 0.0 ) exit( 1 );

const char *transaction_insert_column_list_string(
    const char *TRANSACTION_INSERT,
    const char *TRANSACTION_LOCK_INSERT,
    char transaction_lock_yn );

FILE *transaction_insert_pipe_open(
    const char *transaction_insert_column_list_string(),
    const char *TRANSACTION_TABLE );

char *transaction_race_free_date_time(
        transaction_date_time );

void transaction_insert_pipe(
    FILE *transaction_insert_pipe_open(),
    char *full_name,
    char *street_address,
    char *transaction_race_free_date_time(),
    double transaction_amount,
    char *transaction_check_number(
        int check_number ),
    char *transaction_memo(
        char *memo ),
    char transaction_lock_yn ) );

pclose( transaction_insert_pipe_open() );

if ( insert_journal_list_boolean )
{
    void journal_list_insert(
        full_name,
        street_address,
        race_free_date_time,
        journal_list );

    LIST *journal_extract_account_list( journal_list );

    void journal_account_list_propagate(
        transaction_race_free_date_time(),
        journal_extract_account_list() );
}

/* Usage */
char *transaction_stamp_insert(
    TRANSACTION *transaction /* in/out */,
    boolean insert_journal_list_boolean,
    boolean transaction_lock_boolean );

/* Process */
char transaction_lock_yn(
    transaction_lock_boolean );

char *this->transaction_date_time =
    char *transaction_insert(
        transaction->full_name,
        transaction->street_address,
        transaction->transaction_date_time,
        transaction->transaction_amount,
        transaction->check_number,
        transaction->memo,
        transaction_lock_yn(),
        transaction->journal_list,
        insert_journal_list_boolean );

/* Usage */
void transaction_delete(
    char *full_name,
    char *street_address,
    char * transaction_date_time );

/* Process */
TRANSACTION *transaction =
    transaction_fetch(
        char *full_name,
        char *street_address,
        char *transaction_date_time,
        1 /*  fetch_journal_list */ );

char *transaction_primary_where(
    full_name,
    street_address,
    transaction_date_time );

char *transaction_delete_system_string(
    char *TRANSACTION_TABLE,
    char *transaction_primary_where() );

system( transaction_delete_system_string() );

char *journal_delete_system_string(
    JOURNAL_TABLE,
    transaction_primary_where(
        full_name,
        street_address,
        transaction_date_time ) );

system( journal_delete_system_string() );

void journal_account_list_propagate(
    transaction_date_time,
    journal_extract_account_list(
        transaction->journal_list ) );
```

TRANSACTION (3)

```
/* Usage */
void transaction_fetch_update(
    char *full_name,
    char *street_address,
    char *transaction_date_time );

/* Process */
TRANSACTION *transaction =
    transaction_fetch(
        full_name,
        street_address,
        transaction_date_time,
        1 /* fetch_journal_list */ );

double journal_list_transaction_amount(
    transaction->journal_list );

char *transaction_primary_where(
    full_name,
    street_address,
    transaction_date_time );

void transaction_update(
    journal_list_transaction_amount(),
    transaction_primary_where() );

/* Usage */
void transaction_update(
    double journal_list_transaction_amount(),
    char *transaction_primary_where() );

/* Process */
char *transaction_set_clause(
    char *TRANSACTION_AMOUNT_COLUMN,
    double journal_list_transaction_amount );

char *transaction_update_statement_system_string(
    char *TRANSACTION_TABLE,
    transaction_set_clause(),
    transaction_primary_where );

system(
    transaction_update_statement_system_string() );

/* Usage */
void transaction_subsidiary_date_time_update(
    const char SQL_DELIMITER,
    const char *subsidiary_table_name,
    char *application_name,
    char *primary_key_list_string,
    char *primary_data_list_string,
    const char *date_time_column_name,
    char *date_time_column_value );

/* Process */
char *transaction_date_time_update_statement(
    SQL_DELIMITER,
    subsidiary_table_name,
    primary_key_list_string,
    primary_data_list_string,
    date_time_column_name,
    date_time_column_value );

char *appaserver_error_filename(
    application_name );

char *update_system_string(
    appaserver_error_filename() );

FILE *appaserver_output_pipe(
    update_system_string() );

fprintf(
    appaserver_output_pipe(),
    "%s\n",
    transaction_date_time_update_statement() );

pclose( appaserver_output_pipe() );

free( appaserver_error_filename() );

free( transaction_date_time_update_statement() );

/* Usage */
char *transaction_date_time_update_statement(
    const char SQL_DELIMITER,
    const char *subsidiary_table_name,
    char *primary_key_list_string,
    char *primary_data_list_string,
    const char *date_time_column_name,
    char *date_time_column_value );

/* Process */
LIST *update_column_name_list =
    list_string_list(
        date_time_column_name,
        SQL_DELIMITER );

LIST *update_data_list =
    list_string_list(
        date_time_column_value,
        SQL_DELIMITER );

LIST *primary_key_list =
    list_string_list(
        primary_key_list_string,
        SQL_DELIMITER );

LIST *primary_data_list =
    list_string_list(
        primary_data_list_string,
        SQL_DELIMITER );

char *update_sql_statement_string(
    update_column_name_list,
    update_data_list,
    primary_key_list,
    primary_data_list,
    (LIST *)0 /* folder_attribute_list */,
    subsidiary_table_name );
```

TRANSACTION (4)

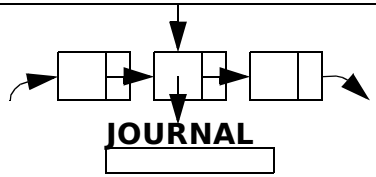
```
/* Usage */
TRANSACTION *transaction_date_time_fetch(
    char *transaction_date_time,
    boolean fetch_journal_list );

/* Process */
char *transaction_date_time_fetch_where(
    transaction_date_time );

char *transaction_system_string(
    TRANSACTION_SELECT,
    TRANSACTION_TABLE,
    transaction_date_time_fetch_where()
       /* where */ );

TRANSACTION *transaction_parse(
    string_pipe_fetch(
       transaction_system_string() )
          /* input */,
    fetch_journal_list );

/* Usage */
void transaction_journal_list_insert(
   LIST *transaction_list,
   boolean with_propagate  );

/* Process */
FILE *journal_insert_pipe(
   JOURNAL_INSERT,
   JOURNAL_TABLE );

for TRANSACTION *transaction in transaction_list
{
   if ( !first_transaction_date_time )
   {
      first_transaction_date_time =
         transaction->transaction_date_time;
   }

   void journal_list_transaction_insert(
      journal_insert_pipe(),
      transaction->full_name,
      transaction->street_address,
      transaction->transaction_date_time,
      transaction->journal_list );
}

pclose( journal_insert_pipe() );

if ( with_propagate )
{
   LIST *transaction_list_extract_account_list(
      transaction_list );

   void journal_account_list_propagate(
      first_transaction_date_time,
      transaction_list_extract_account_list() );
}

/* Usage */
LIST *transaction_list_extract_account_list(
   LIST *transaction_list );

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

for TRANSACTION *transaction in transaction_list
{
   void journal_account_list_getset(
      account_list /* in/out */,
      transaction->journal_list );
}

/* Usage */
void transaction_list_html_display(
   LIST *transaction_list );

/* Process */
for TRANSACTION *transaction in transaction_list
{
   void transaction_html_display( transaction );
}

/* Usage */
void transaction_html_display(
   TRANSACTION *transaction );

/* Process */
void journal_list_html_display(
   transaction->journal_list,
   transaction->transaction_date_time,
   transaction->transaction_memo,
   transaction->full_name );

/* Usage */
TRANSACTION *transaction_binary_account_key(
   const char *debit_account_key,
   const char *credit_account_key,
   char *full_name,
   char *street_address,
   char *transaction_date_time,
   double transaction_amount,
   char *memo );

/* Process */
if ( !transaction_date_time || !transaction_amount )
      return NULL;

char *debit_account_name =
   account_hard_coded_account_name(
      debit_account_key,
      0 /* not warning_only */,
      __FUNCTION__ /* calling_function_name */ );

char *credit_account_name =
   account_hard_coded_account_name(
      credit_account_key,
      0 /* not warning_only */,
      __FUNCTION__ /* calling_function_name */ );

TRANSACTION *transaction =
   transaction_binary(
      full_name,
      street_address,
      transaction_date_time,
      transaction_amount,
      memo,
      debit_account_name,
      credit_account_name );
```

TRANSACTION (5)

```
/* Usage */
char *transaction_refresh(
    char *full_name,
    char *street_address,
    char *transaction_date_time,
    double transaction_amount,
    int check_number,
    char *memo,
    char lock_transaction_yn,
    LIST *journal_list );

/* Process */
void transaction_delete(
    full_name,
    street_address,
    transaction_date_time );

char *transaction_date_time =
    char *transaction_insert(
        full_name,
        street_address,
        transaction_date_time,
        transaction_amount,
        check_number,
        memo,
        lock_transaction_yn,
        journal_list,
        1 /* insert_journal_list */ );

/* Usage */
/* Increments second each invocation. */
char *transaction_increment_date_time(
    char *transaction_date_string );

/* Process */
static DATE *static_date_second = {0};

if (    !transaction_date_string
||      !*transaction_date_string
||      strcmp(
            transaction_date_string,
            "transaction_date ) == 0 )
{
    transaction_date_string =
        char *date_now_yyyy_mm_dd(
            date_utc_offset() );
}

if ( !static_date_second )
{
    char *date_now_hhmmss( utc_offset() );

    static_date_second =
        DATE *date_yyyy_mm_dd_hhmmss_new(
            transaction_date_string,
            date_now_hhmmss() );
}

char *date_hhmmss( static_date_second );

sprintf(
    transaction_date_time[],
    "%s %s",
    transaction_date_string,
    date_hhmmss() );

free( date_hhmmss() );

void date_increment_second(
    static_date_second,
    1 /* second */ );

char *strdup( transaction_date_time );
```

TRANSACTION (6)

```
/* Usage */
char *transaction_race_free_date_time(
    char *transaction_date_time );

/* Process */
DATE *next_transaction_date_time = {0};

SEMAPHORE *semaphore =
    semaphore_new(
       TRANSACTION_SEMAPHORE_KEY );

void semaphore_wait( semaphore->id );

while ( transaction_date_time_boolean(
            TRANSACTION_TABLE,
            TRANSACTION_DATE_TIME_COLUMN,
            transaction_date_time ) )
{
    if ( !next_transaction_date_time )
    {
       next_transaction_date_time =
          date_yyyy_mm_dd_hms_new(
             transaction_date_time );
    }

    date_increment_second(
       next_transaction_date_time,
       1 );

    transaction_date_time =
       date_display_yyyy_mm_dd_colon_hms(
          next_transaction_date_time );
}

semaphore_signal( semaphore->id );

return transaction_date_time;

/* Usage */
char *transaction_primary_where(
    char *full_name,
    char *street_address,
    char *transaction_date_time );

/* Process */

/* Usage */
char *transaction_system_string(
    char *TRANSACTION_SELECT,
    char *TRANSACTION_TABLE,
    char *where );

/* Process */

/* Attributes */
char *full_name;
char *street_address;
char *transaction_date_time;

/* Set externally */
double transaction_amount;
int check_number;
char *memo;
LIST *journal_list;
char *rental_property_street_address;
char *fund_name;
```



TRANSACTION_DATE (1)

```
/* Usage */
char *transaction_date_time_string(
    char *date_string,
    char *time_string );

/* Process */
sprintf(
    static date_time_string[],
    "%s %s",
    date_string,
    time_string );

/* Usage */
char *transaction_date_end_date_time_string(
    char *end_date_string,
    char *end_time_string );

/* Process */
char *transaction_date_time_string(
    end_date_string,
    end_time_string );

char *strdup( transaction_date_time_string() );

/* Usage */
char *transaction_date_as_of(
    const char *TRANSACTION_TABLE,
    char *as_of_date_string );

/* Process */
boolean transaction_date_as_of_populated(
    char *as_of_date_string );

if ( transaction_date_as_of_populated() )
{
    return as_of_date_string;
}
else
{
    char *transaction_date_time_maximum_string(
        TRANSACTION_TABLE,
        (char *)0 /* where_string */ );

    if ( !transaction_date_time_maximum_string() )
        return NULL;

    char *column(
        date_as_of[],
        0,
        transaction_date_time_maximum_string() );
}

/* Usage */
boolean transaction_date_close_boolean(
    const char *TRANSACTION_TABLE,
    const char *TRANSACTION_DATE_CLOSE_TIME,
    const char *TRANSACTION_CLOSE_MEMO,
    char *transaction_date_as_of() );

/* Process */
char *transaction_date_close_where_string(
    TRANSACTION_DATE_MEMO,
    transaction_date_as_of /* date_string */,
    TRANSACTION_DATE_CLOSE_TIME /* time_string */ );

char *transaction_date_count_system_string(
    TRANSACTION_TABLE,
    transaction_date_close_where_string() );

char *string_pipe(
    transaction_date_count_system_string() );

return ( atoi( string_pipe() == 1 );

/* Usage */
char *transaction_date_close_where_string(
    const char *TRANSACTION_CLOSE_MEMO,
    char *date_string,
    char *time_string );

/* Process */
char *transaction_date_time_string(
    date_string,
    time_string );

sprintf(
    static where_string[],
    "transaction_date_time = '%s' and "
    "memo = '%s'",
    transaction_date_time_string(),
    TRANSACTION_CLOSE_MEMO );

/* Usage */
char *transaction_date_begin_date_string(
    char *TRANSACTION_TABLE,
    char *transaction_date_as_of() );

/* Process */
DATE *transaction_date_prior_close_date(
    TRANSACTION_DATE_CLOSE_TIME,
    TRANSACTION_CLOSE_MEMO,
    TRANSACTION_TABLE,
    transaction_date_as_of );

if ( transaction_date_prior_close_date() )
{
    date_increment_days(
        transaction_date_prior_close_date(),
        1.0 );

    char *date_yyyy_mm_dd_string(
        transaction_date_prior_close_date() );
}
else
{
    return
    transaction_date_minimum_string(
        TRANSACTION_TABLE );
}
```

TRANSACTION_DATE (2)

```
/* Usage */
char *transaction_date_close_date_time_string(
    const char *TRANSACTION_DATE_PRECLOSE_TIME,
    const char *TRANSACTION_DATE_CLOSE_TIME,
    char *transaction_date_as_of(),
    boolean preclose_time_boolean );

/* Process */
if ( preclose_time_boolean )
(
    char *transaction_date_time_string(
       transaction_date_as_of,
       TRANSACTION_DATE_PRECLOSE_TIME );
}
else
{
    char *transaction_date_time_string(
       transaction_date_as_of,
       TRANSACTION_DATE_CLOSE_TIME );
}

char *strdup( transaction_date_time_string() );

/* Usage */
DATE *transaction_date_prior_close_date(
    const char *TRANSACTION_TABLE,
    const char *TRANSACTION_DATE_CLOSE_TIME,
    const char *TRANSACTION_CLOSE_MEMO,
    char *transaction_date_as_of() );

/* Process */
char *select = “max( transaction_date_time )”;

char *transaction_date_time_string(
    transaction_date_as_of
       /* date_string */,
    TRANSACTION_DATE_CLOSE_TIME
       /* time_string */ );

sprintf(
    where_string[],
    “memo = ‘%s’ and transaction_date_time < ‘%s’”,
    TRANSACTION_CLOSE_MEMO,
    transaction_date_time_string() );

sprintf(
    system_string[],
    “select.sh \”%s\” %s \”%s\””,
    select,
    TRANSACTION_TABLE,
    where_string );

char *string_pipe( system_string );

if ( string_pipe() )
{
    DATE *date_yyyy_mm_dd_new(
       string_pipe_fetch() );
}
else
{
    return NULL;
}

/* Usage */
boolean transaction_date_time_changed(
    char *preupdate_transaction_date_time );

/* Process */
if ( !preupdate_transaction_date_time ) return 0;
if ( !*preupdate_transaction_date_time ) return 0;

if ( strcmp(
       preupdate_transaction_date_time,
       “preupdate_transaction_date_time” ) == 0 )
{
    return 0;
}
return 1;

/* Usage */
char *transaction_earlier_date_time(
    char *transaction_date_time,
    char *preupdate_transaction_date_time );

/* Process */
boolean transaction_date_time_changed(
    preupdate_transaction_date_time );

if ( !transaction_date_time_changed() )
{
    earlier_date_time = transaction_date_time;
}
else
{
    int string_strcmp(
          transaction_date_time,
          preupdate_transaction_date_time );

    if ( string_strcmp() <= 0 )
       date_time_earlier =
          transaction_date_time;
    else
       date_time_earlier =
          preupdate_transaction_date_time;
}

/* Usage */
boolean transaction_date_time_boolean(
    const char *TRANSACTION_TABLE,
    char *transaction_date_time );

/* Process */
char *transaction_date_time_where_string(
    char *transaction_date_time );

char *transaction_date_count_system_string(
    TRANSACTION_TABLE,
    transaction_date_time_where_string() );

char *string_pipe(
    transaction_date_count_system_string() );

return ( atoi( string_pipe() == 1 );
```

TRANSACTION_DATE (3)

```
/* Usage */
char *transaction_date_minimum_string(
    const char *TRANSACTION_TABLE );

/* Process */
char *select = “min( transaction_date_time )”;

sprintf(
    system_string[],
    “select.sh \”%s\” %s”,
    select,
    TRANSACTION_TABLE );

char *string_pipe( system_string );

if ( string_pipe() )
{
    char *column(
        minimum_string[],
        0,
        string_pipe() );
}
else
{
    return NULL;
}

/* Usage */
char *transaction_date_time_maximum_string(
    const char *TRANSACTION_TABLE,
    char *where_string );

/* Process */
char *select = “max( transaction_date_time )”;

sprintf(
    system_string[],
    “select.sh \”%s\” %s \”%s\””,
    select,
    TRANSACTION_TABLE,
    where_string );

char *string_pipe( system_string );

/* Usage */
char *transaction_date_time_memo_maximum_string(
    const char *TRANSACTION_TABLE,
    const char *TRANSACTION_CLOSE_MEMO,
    char *entity_self_full_name,
    char *entity_self_street_address );

/* Process */
char *entity_primary_where(
    entity_self_full_name,
    entity_self_street_address );

sprintf(
    where_string[],
    “%s and memo = ‘%s’”,
    entity_primary_where(),
    TRANSACTION_CLOSE_MEMO );

char *transaction_date_time_maximum_string(
    TRANSACTION_TABLE,
    where_string );

/* Usage */
char *transaction_date_count_system_string(
    const char *TRANSACTION_TABLE,
    char *where_string );

/* Process */
char *select = “count(1)”;

sprintf(
    system_string[],
    “select.sh \”%s\” %s \”%s\””,
    select,
    TRANSACTION_TABLE,
    where_string );

/* Usage */
char *transaction_date_prior_end_date_time_string(
    char *transaction_date_begin_date_time_string );

/* Process */
DATE *prior =
    date_yyyy_mm_dd_new(
        transaction_date_begin_date_time_string );

date_decrement_second( prior, 1 );

char *date_display19( prior );
```

TRANSACTION_DATE_TRIAL_BALANCE

```
/* Usage */
TRANSACTION_DATE_TRIAL_BALANCE *
    transaction_date_trial_balance_new(
        char *as_of_date_string );

/* Process */
TRANSACTION_DATE_TRIAL_BALANCE *
    transaction_date_trial_balance_calloc(
        void );

char *transaction_date_as_of(
    TRANSACTION_TABLE,
    as_of_date_string );

if ( !transaction_date_as_of() ) return this;

char *transaction_date_begin_date_string(
    TRANSACTION_TABLE,
    transaction_date_as_of() );

boolean transaction_date_close_boolean(
    TRANSACTION_TABLE,
    TRANSACTION_DATE_CLOSE_TIME,
    TRANSACTION_CLOSE_MEMO,
    transaction_date_as_of() );

if ( transaction_date_close_boolean() )
{
    char *preclose_end_date_time_string =
        transaction_date_close_date_time_string(
            TRANSACTION_DATE_PRECLOSE_TIME,
            TRANSACTION_DATE_CLOSE_TIME,
            transaction_date_as_of(),
            1 /* preclose_time_boolean */ );
}

char *end_date_time_string =
    transaction_date_close_date_time_string(
        TRANSACTION_DATE_PRECLOSE_TIME,
        TRANSACTION_DATE_CLOSE_TIME,
        transaction_date_as_of(),
        0 /* not preclose_time_boolean */ );

/* Attributes */
char *transaction_date_as_of;
char *transaction_date_begin_date_string;
boolean transaction_date_close_boolean;
char *preclose_end_date_time_string;
char *end_date_time_string;
```

TRANSACTION_DATE_STATEMENT

```
/* Usage */
TRANSACTION_DATE_STATEMENT *
    transaction_date_statement_new(
        char *as_of_date_string );

/* Process */
TRANSACTION_DATE_STATEMENT *
    transaction_date_statement_calloc(
        void );

char *transaction_date_as_of(
    TRANSACTION_TABLE,
    as_of_date_string );

if ( !transaction_date_as_of() ) return this;

char *transaction_date_begin_date_string(
    TRANSACTION_TABLE,
    transaction_date_as_of() );

if ( !transaction_date_begin_date_string() exit( 1 );

boolean transaction_date_close_boolean(
    TRANSACTION_TABLE,
    TRANSACTION_DATE_CLOSE_TIME,
    TRANSACTION_CLOSE_MEMO,
    transaction_date_as_of() );

char *end_date_time_string =
    transaction_date_close_date_time_string(
        TRANSACTION_DATE_PRECLOSE_TIME,
        TRANSACTION_DATE_CLOSE_TIME,
        transaction_date_as_of(),
        transaction_date_close_boolean()
            /* preclose_time_boolean */ );

char *prior_end_date_time_string =
    transaction_date_prior_end_date_time_string(
        transaction_date_begin_date_string );

/* Attributes */
char *transaction_date_as_of;
char *transaction_date_begin_date_string;
boolean transaction_date_close_boolean;
char *end_date_time_string;
char *prior_end_date_time_string;
```

TRANSACTION_DATE_CLOSE_NOMINAL_DO

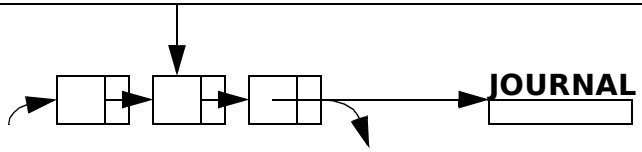
```
/* Usage */
TRANSACTION_DATE_CLOSE_NOMINAL_DO *
    transaction_date_close_nominal_do_new(
        char *as_of_date_string );

/* Process */
TRANSACTION_DATE_CLOSE_NOMINAL_DO *
    transaction_date_close_nominal_do_calloc(
        void );

boolean transaction_date_close_boolean(
    TRANSACTION_TABLE,
    TRANSACTION_DATE_CLOSE_TIME,
    TRANSACTION_CLOSE_MEMO,
    as_of_date_string );

char *transaction_date_close_date_time(
    TRANSACTION_DATE_PRECLOSE_TIME,
    TRANSACTION_DATE_CLOSE_TIME,
    as_of_date_string,
    0 /* not preclose_time_boolean */ );

/* Attributes */
boolean transaction_date_close_boolean;
char *transaction_date_close_date_time;
```

TRANSACTION_DATE_CLOSE_NOMINAL_UNDO

```
/* Usage */
TRANSACTION_DATE_CLOSE_NOMINAL_UNDO *
    transaction_date_close_nominal_undo_new(
        char *entity_self_full_name,
        char *entity_self_street_address );

/* Process */
TRANSACTION_DATE_CLOSE_NOMINAL_UNDO *
    transaction_date_close_nominal_undo_calloc(
        void );

char *transaction_date_time_memo_maximum_string(
    TRANSACTION_TABLE,
    TRANSACTION_CLOSE_MEMO,
    entity_self_full_name,
    entity_self_street_address );

/* Attributes */
char *transaction_date_time_memo_maximum_string;
```

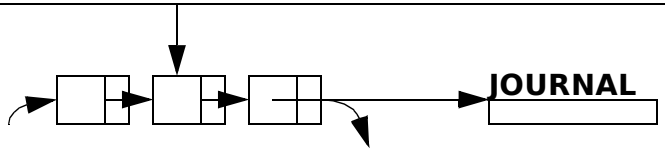
TRANSACTION_PROGRAM

```
/* Usage */
TRANSACTION *transaction_program_fetch(
    full_name,
    street_address,
    transaction_date_time,
    char *program_name,
    boolean fetch_journal_ledger );

/* Process */
```



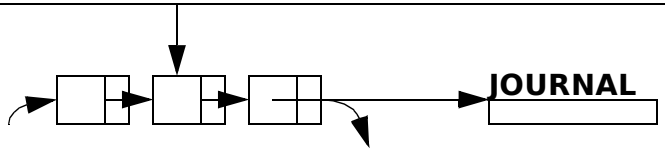
TRANSACTION_FUND

```
/* Usage */
TRANSACTION *transaction_program_fetch(
    full_name,
    street_address,
    transaction_date_time,
    char *fund_name,
    boolean fetch_journal_ledger );

/* Process */
```



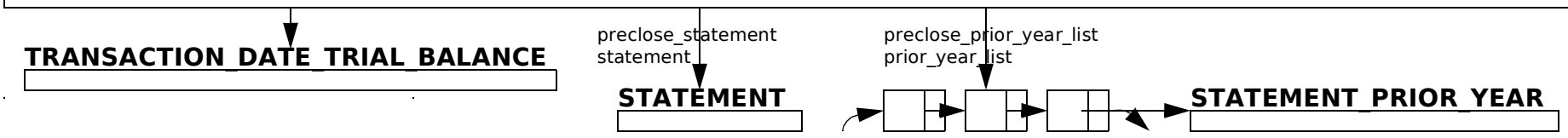
TRANSACTION_PROPERTY

```
/* Usage */
TRANSACTION *transaction_property_fetch(
    full_name,
    street_address,
    transaction_date_time,
    char *property_street_address,
    boolean fetch_journal_ledger );

/* Process */
```



TRIAL_BALANCE (1)

```
/* Usage */
TRIAL_BALANCE *trial_balance_fetch(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *process_name,
    char *appaserver_parameter_data_directory,
    char *as_of_date_string,
    int prior_year_count,
    char *subclassification_option_string,
    char *output_medium_string );

/* Process */
enum statement_subclassification_option =
    statement_resolve_subclassification_option(
        subclassification_option_string );

enum statement_output_medium =
    statement_resolve_output_medium(
        output_medium_string );

TRANSACTION_DATE_TRIAL_BALANCE *
    transaction_date_trial_balance =
        transaction_date_trial_balance_new(
            as_of_date_string );

if ( !transaction_date_trial_balance->
        transaction_date_as_of )
{
    return NULL;
}

LIST *trial_balance_element_name_list(
    char *ELEMENT_ASSET,
    char *ELEMENT_LIABILITY,
    char *ELEMENT_REVENUE,
    char *ELEMENT_EXPENSE,
    char *ELEMENT_GAIN,
    char *ELEMENT_LOSS,
    char *ELEMENT_EQUITY );

if ( transaction_date_trial_balance->
        transaction_date_close_boolean )
{
    char *trial_balance_preclose_process_name(
        char *process_name );

    STATEMENT *preclose_statement =
        statement_fetch(
            application_name,
            trial_balance_preclose_process_name(),
            prior_year_count,
            trial_balance_element_name_list(),
            transaction_date_trial_balance->
                transaction_date_begin_date_string,
            transaction_date_trial_balance->
                preclose_end_date_time_string,
            1 /* fetch_transaction */ );

    if ( !preclose_statement ) exit( 1 );

    void element_account_transaction_count_set(
        preclose_statement->element_statement_list,
        transaction_date_trial_balance->
            transaction_date_begin_date_string,
        transaction_date_trial_balance->
            preclose_end_date_time_string );

    if ( statement_output_medium == output_table )
    {
        void element_account_action_string_set(
            preclose_statement->
                element_statement_list,
            application_name,
            session_key,
            login_name,
            role_name,
            transaction_date_trial_balance->
                transaction_date_begin_date_string,
            transaction_date_trial_balance->
                preclose_end_date_time_string );
    }

    if ( prior_year_count )
    {
        LIST *preclose_statement_prior_year_list =
            LIST *statement_prior_year_list(
                trial_balance_element_name_list(),
                transaction_date_trial_balance->
                    preclose_end_date_time_string,
                prior_year_count,
                preclose_statement );
    }
}

STATEMENT *statement =
    statement_fetch(
        application_name,
        process_name,
        prior_year_count,
        trial_balance_element_name_list(),
        transaction_date_trial_balance->
            transaction_date_begin_date_string,
        transaction_date_trial_balance->
            end_date_time_string,
        1 /* fetch_transaction */ );

if ( !statement ) exit( 1 );
```



TRIAL_BALANCE (2)

```
/* Process (continued) */
void element_account_transaction_count_set(
   statement->element_statement_list,
   transaction_date_trial_balance->
      transaction_date_begin_date_string,
   transaction_date_trial_balance->
      end_date_time_string );

if ( statement_output_medium ==
      statement_output_table )
{
   void element_account_action_string_set(
      statement->element_statement_list,
      application_name,
      session_key,
      login_name,
      role_name,
      transaction_date_trial_balance->
         transaction_date_begin_date_string,
      transaction_date_trial_balance->
         end_date_time_string );
}

if ( prior_year_count )
{
   LIST *statement_prior_year_list(
      trial_balance_element_name_list(),
      transaction_date_trial_balance->
         end_date_time_string,
      prior_year_count,
      statement );
}

if ( transaction_date_trial_balance->
      transaction_date_close_boolean )
{
   double preclose_debit_sum =
      double element_list_debit_sum(
         preclose_statement->
            element_statement_list );

   double preclose_credit_sum =
      double element_list_credit_sum(
         preclose_statement->
            element_statement_list );
}

double debit_sum =
   double element_list_debit_sum(
      statement->
         element_statement_list );

double credit_sum =
   double element_list_credit_sum(
      statement->
         element_statement_list );

if (   transaction_date_trial_balance->
         transaction_date_close_boolean
&&  statement_subclassification_option ==
         statement_subclassification_omit )
{
   element_list_account_statement_list_set(
      preclose_statement->
         element_statement_list );
}

if (   statement_subclassification_option ==
         statement_subclassification_omit )
{
   element_list_account_statement_list_set(
      statement->
         element_statement_list );
}

if ( statement_output_medium ==
      statement_output_PDF )
{
   TRIAL_BALANCE_PDF *trial_balance_pdf =
      trial_balance_pdf_new(
         application_name,
         appaserver_parameter_data_directory,
         statement_subclassification_option,
         preclose_statement,
         preclose_statement_prior_year_list,
         preclose_debit_sum,
         preclose_credit_sum,
         statement,
         statement_prior_year_list,
         debit_sum,
         credit_sum,
         getpid() /* process_id */ );
}
else
if ( statement_output_medium ==
      statement_output_table )
{
   TRIAL_BALANCE_TABLE *trial_balance_table =
      trial_balance_table_new(
         application_name,
         login_name,
         statement_subclassification_option,
         preclose_statement,
         preclose_statement_prior_year_list,
         preclose_debit_sum,
         preclose_credit_sum,
         statement,
         statement_prior_year_list,
         debit_sum,
         credit_sum );
}

/* Usage */
char *trial_balance_transaction_count_string(
   int transaction_count );

/* Process */

/* Attributes */
enum statement_subclassification_option
   statement_subclassification_option;
enum statement_output_medium
   statement_output_medium;
TRANSACTION_DATE_TRIAL_BALANCE *
   transaction_date_trial_balance;
LIST *element_name_list;
STATEMENT *preclose_statement;
LIST *preclose_statement_prior_year_list;
STATEMENT *statement;
LIST *statement_prior_year_list;
double preclose_debit_sum;
double preclose_credit_sum;
double debit_sum;
double credit_sum;
TRIAL_BALANCE_PDF *trial_balance_pdf;
TRIAL_BALANCE_TABLE *trial_balance_table;
```

TRIAL_BALANCE_PDF **TRIAL_BALANCE_TABLE**

TRIAL_BALANCE (3)

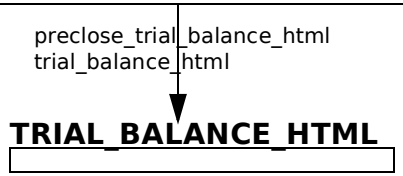
```
/* Driver */
printf(
    "%s\n",
    trial_balance->
       statement->
       statement_caption->
       frame_title );

if ( trial_balance->trial_balance_pdf )
{
    if ( trial_balance->
           trial_balance_pdf->
           preclose_trial_balance_latex )
    {
       void latex_table_output(
          trial_balance->
             trial_balance_pdf->
             preclose_statement_link->
             tex_filename,
          trial_balance->
             preclose_statement_link->
             appaserver_link_working_directory,
          trial_balance->
             trial_balance_pdf->
             preclose_statement_link->
             pdf_anchor_html,
          trial_balance->
             trial_balance_pdf->
             preclose_trial_balance_latex->
             latex->
          trial_balance->
             trial_balance_pdf->
             preclose_trial_balance_latex->
             latex_table );
}

void latex_table_output(
   trial_balance->
      trial_balance_pdf->
      statement_link->
      tex_filename,
   trial_balance->
      trial_balance->
      trial_balance_pdf->
      statement_link->
      appaserver_link_working_directory,
   trial_balance->
      trial_balance_pdf->
      statement_link->
      pdf_anchor_html,
   trial_balance->
      trial_balance_pdf->
      trial_balance_latex->
      latex->
   trial_balance->
      trial_balance_pdf->
             trial_balance_latex->
             latex_table );
}
else
if ( trial_balance->trial_balance_table )
{
    if ( trial_balance->
           trial_balance_table->
           preclose_trial_balance_html )
    {
       void statement_html_output(
          trial_balance->
             trial_balance_table->
             preclose_trial_balance_html->
             html_table,
          TRIAL_BALANCE_PRECLOSE_TITLE
             /* secondary_title */ );
    }

    void statement_html_output(
       trial_balance->
          trial_balance_table->
          trial_balance_html->
          html_table,
       (char *)0 /* secondary_title */ );
}
```

TRIAL_BALANCE_TABLE

```
/* Usage */
TRIAL_BALANCE_TABLE *trial_balance_table_new(
    char *application_name,
    char *login_name,
    enum statement_subclassification_option
       statement_subclassification_option,
    char *transaction_as_of_date,
    STATEMENT *preclose_statement,
    LIST *preclose_statement_prior_year_list,
    double preclose_debit_sum,
    double preclose_credit_sum,
    STATEMENT *statement,
    LIST *statement_prior_year_list,
    double debit_sum,
    double credit_sum );
```

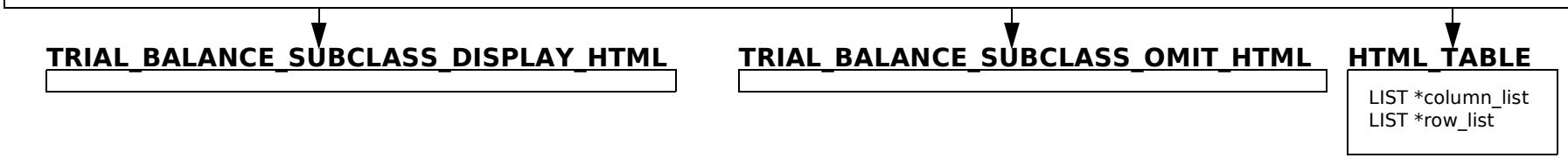
```
/* Process */
TRIAL_BALANCE_TABLE *trial_balance_table_calloc(
    void );

if ( preclose_statement )
{
    TRIAL_BALANCE_HTML *
       preclose_trial_balance_html =
          trial_balance_html_new(
             application_name,
             login_name,
             statement_subclassification_option,
             preclose_statement,
             preclose_statement_prior_year_list,
             preclose_debit_sum,
             preclose_credit_sum );
```

```
}

TRIAL_BALANCE_HTML *trial_balance_html =
    trial_balance_html_new(
       application_name,
       login_name,
       statement_subclassification_option,
       statement,
       statement_prior_year_list,
       debit_sum,
       credit_sum );

/* Attributes */
TRIAL_BALANCE_HTML *preclose_trial_balance_html;
TRIAL_BALANCE_HTML *trial_balance_html;
```



TRIAL_BALANCE_HTML (1)

```
/* Usage */
TRIAL_BALANCE_HTML *trial_balance_html_new(
    char *application_name,
    char *login_name,
    enum statement_subclassification_option
       statement_subclassification_option,
    STATEMENT *statement,
    LIST *statement_prior_year_list,
    double debit_sum,
    double credit_sum );

/* Process */
TRIAL_BALANCE_HTML *trial_balance_html_calloc(
    void );

if (statement_subclassification_option ==
    subclassification_display )
{
    TRIAL_BALANCE_SUBCLASS_DISPLAY_HTML *
       trial_balance_subclass_display_html =
          trial_balance_subclass_display_html_new(
             application_name,
             login_name,
             statement,
             statement_prior_year_list,
             debit_sum,
             credit_sum );

    HTML_TABLE *trial_balance_html_table(
       statement->statement_caption->sub_title,
       trial_balance_subclass_display_html->
          column_list,
       trial_balance_subclass_display_html->
          row_list );
}
else
{
    TRIAL_BALANCE_SUBCLASS_OMIT_HTML *
       trial_balance_subclass_omit_html =
          trial_balance_subclass_omit_html_new(
             statement,
             statement_prior_year_list,
             debit_sum,
             credit_sum );

    HTML_TABLE *trial_balance_html_table(
       statement->statement_caption->sub_title,
       trial_balance_subclass_omit_html->
          column_list,
       trial_balance_subclass_omit_html->
          row_list );
}

/* Usage */
HTML_TABLE *trial_balance_html_table(
    char *statement_caption_sub_title,
    LIST *column_list,
    LIST *row_list );

/* Process */
HTML_TABLE *html_table =
    html_table_new(
       (char *)0 /* title */,
       statement_caption_sub_title,
       (char *)0 /* sub_sub_title */ );
html_table->column_list = column_list;
html_table->row_list = row_list;

/* Usage */
char *trial_balance_html_account_datum(
    char *account->account_name,
    char *account->
       account_journal_latest->
       full_name,
    double account->
       account_journal_latest->
       debit_amount,
    double account->
       account_journal_latest->
       credit_amount,
    char *statement_date_american(
       account->
          account_journal_latest->
          transaction_date_time )
             /* transaction_date_american */,
    char *account->
       account_journal_latest->
       transaction->
       memo );

/* Process */

/* Attributes */
TRIAL_BALANCE_SUBCLASS_DISPLAY_HTML *
    trial_balance_subclass_display_html;
TRIAL_BALANCE_SUBCLASS_OMIT_HTML *
    trial_balance_subclass_omit_html;
HTML_TABLE *html_table;
```

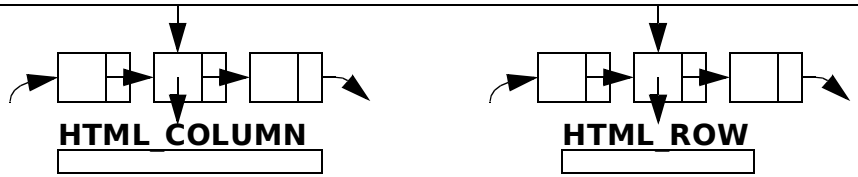


TRIAL_BALANCE_HTML (2)

```
/* Usage */
HTML_ROW *trial_balance_html_total_row(
    double debit_sum,
    double credit_sum,
    int column_skip_count );

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

list_set(
    cell_list,
    html_cell_new(
       “Total”,
       0 /* not large_boolean */,
       0 /* not bold_boolean */ ) );

for ( i = 0; i < column_skip_count; i++ )
{
    list_set(
       cell_list,
       html_cell_new(
          (char *)0 /* datum */,
          0 /* not large_boolean */,
          0 /* not bold_boolean */ ) );
}

char *string_commas_double(
    debit_sum,
    2 /* decimal_place_count */ );

list_set(
    cell_list,
    html_cell_new(
       strdup( string_commas_double() ) /* datum */,
          0 /* not large_boolean */,
          0 /* not bold_boolean */ ) );

char *string_commas_double(
    credit_sum,
    2 /* decimal_place_count */ );

list_set(
    cell_list,
    html_cell_new(
       strdup( string_commas_double() ) /* datum */,
       0 /* not large_boolean */,
       0 /* not bold_boolean */ ) );

HTML_ROW *html_row_new( cell_list );
```

TRIAL_BALANCE_SUBCLASS_DISPLAY_HTML (1)

```
/* Usage */
TRIAL_BALANCE_SUBCLASS_DISPLAY_HTML *
    trial_balance_subclass_display_html_new(
        char *application_name,
        char *login_name,
        STATEMENT *statement,
        LIST *statement_prior_year_list(),
        double debit_sum,
        double credit_sum );

/* Process */
TRIAL_BALANCE_SUBCLASS_DISPLAY_HTML *
    trial_balance_subclass_display_html_calloc(
        void );

LIST *trial_balance_subclass_display_html_column_list(
    statement_prior_year_list );

LIST *row_list = list_new();

for ELEMENT *element in
    statement->element_statement_list
{
    if ( !element->sum ) continue;

    char *element_name = element->element_name;

    for SUBCLASSIFICATION *subclassification in
        element->subclassification_statement_list
    {
        if ( !subclassification->sum ) continue;

        LIST *trial_balance_subclass_display_html_row_list(
            application_name,
            login_name,
            statement->end_date_time_string,
            element_name,
            element->accumulate_debit,
            subclassification->subclassification_name,
            subclassification->account_statement_list,
            statement_prior_year_list );

        list_set_list(
            row_list,
            trial_balance_subclass_display_html_row_list() );

        element_name = (char *)0;
    }
}

HTML_ROW *trial_balance_html_total_row(
    debit_sum,
    credit_sum,
    3 /* column_skip_count */ );

list_set(
    row_list,
    trial_balance_html_total_row() );

/* Attributes */
LIST *column_list;
LIST *row_list;
```



TRIAL_BALANCE_SUBCLASS_DISPLAY_HTML (2)

```
/* Usage */
LIST *trial_balance_subclass_display_html_column_list(
    LIST *statement_prior_year_list );

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

list_set(
    column_list,
    html_column_new(
        STATEMENT_ELEMENT_HEADING,
        0 /* not right_justify_boolean */ ) );

list_set(
    column_list,
    html_column_new(
        STATEMENT_SUBCLASSIFICTION_HEADING,
        0 /* not right_justify_boolean */ ) );

list_set(
    column_list,
    html_column_new(
        STATEMENT_ACCOUNT_HEADING,
        0 /* not right_justify_boolean */ ) );

list_set(
    column_list,
    html_column_new(
        STATEMENT_COUNT_HEADING,
        1 /* right_justify_boolean */ ) );

list_set(
    column_list,
    html_column_new(
        STATEMENT_DEBIT_HEADING,
        1 /* right_justify_boolean */ ) );

list_set(
    column_list,
    html_column_new(
        STATEMENT_CREDIT_HEADING,
        1 /* right_justify_boolean */ ) );

list_set(
    column_list,
    html_column_new(
        STATEMENT_PERCENT_OF_ASSET_HEADING,
        1 /* right_justify_boolean */ ) );

list_set(
    column_list,
    html_column_new(
        STATEMENT_PERCENT_OF_INCOME_HEADING,
        1 /* right_justify_boolean */ ) );

if ( list_length( statement_prior_year_heading_list ) )
{
    LIST *statement_prior_year_heading_list(
        statement_prior_year_list );

    LIST *html_column_right_list(
        statement_prior_year_heading_list() );

    list_set_list(
        column_list,
        html_column_right_list() );
}
```

TRIAL_BALANCE_SUBCLASS_DISPLAY_HTML (3)

```
/* Usage */
LIST *trial_balance_subclass_display_html_row_list(
    char *application_name,
    char *login_name,
    char *statement->end_date_time_string,
    char *element_name,
    boolean element_accumulate_debit,
    char *subclassification_name,
    LIST *account_statement_list,
    LIST *statement_prior_year_list );

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

for ACCOUNT *account in account_statement_list
{
   STATEMENT_ACCOUNT *statement_account =
      statement_account_new(
         end_date_time_string,
         element_accumulate_debit,
         account->account_journal_latest,
         account->action_string,
         0 /* not round_dollar_boolean */,
         account );

   HTML_ROW *
      trial_balance_subclass_display_html_row(
         application_name,
         login_name,
         element_name,
         subclassification_name,
         statement_account,
         statement_prior_year_list ) );

   list_set(
      row_list,
      trial_balance_subclass_display_html_row() );

   element_name = (char *)0;
   subclassification_name = (char *)0;
}
```

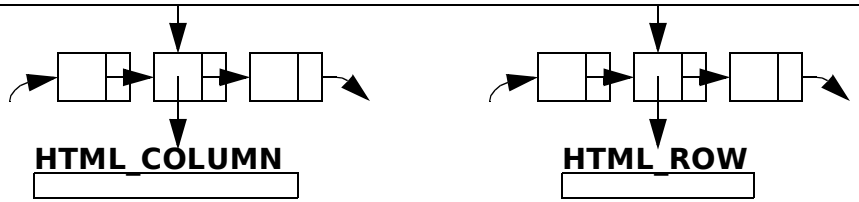
TRIAL_BALANCE_SUBCLASS_DISPLAY_HTML (4)

```
/* Usage */
HTML_ROW *trial_balance_subclass_display_html_row(
    char *application_name,
    char *login_name,
    char *element_name,
    char *subclassification_name,
    STATEMENT_ACCOUNT *statement_account,
    LIST *statement_prior_year_list );

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

char *statement_cell_label_datum(
    element_name );

list_set(
    cell_list,
    html_cell_new(
        statement_cell_label_datum(),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

char *statement_cell_label_datum(
    subclassification_name );

list_set(
    cell_list,
    html_cell_new(
        statement_cell_label_datum(),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

char *transaction_date_convert =
    char *statement_date_convert(
        application_name,
        login_name,
        statement_account->
            account->
            account_journal_latest->
            transaction_date_time );

char *trial_balance_html_account_datum(
    statement_account->
        account->
        account_name,
    statement_account->
        account->
        account_journal_latest->
        full_name,
    statement_account->
        account->
        account_journal_latest->
        debit_amount,
    statement_account->
        account->
        account_journal_latest->
        credit_amount,
    transaction_date_convert,
    statement_account->
        account->
        account_journal_latest->
        transaction->memo );

list_set(
    cell_list,
    html_cell_new(
        trial_balance_html_account_datum(),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

char *trial_balance_transaction_count_string(
    statement_account->account->transaction_count );

list_set(
    cell_list,
    html_cell_new(
        trial_balance_transaction_count_string(),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        statement_account->debit_string,
        0 /* not large_boolean */,
        statement_account->
            within_days_between_boolean
            /* bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        statement_account->credit_string,
         0 /* not large_boolean */,
        statement_account->
            within_days_between_boolean
            /* bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        statement_account->asset_percent_string,
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        statement_account->revenue_percent_string,
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

if ( list_length( statement_prior_year_list ) )
{
    LIST *statement_prior_year_account_data_list(
        statement_account->account->account_name,
        statement_prior_year_list );

    LIST *html_cell_list(
        statement_prior_year_account_data_list() );

    list_set_list(
        cell_list,
        html_cell_list() );
}
```

TRIAL_BALANCE_SUBCLASS_OMIT_HTML (1)

```
/* Usage */
TRIAL_BALANCE_SUBCLASS_OMIT_HTML *
    trial_balance_subclass_omit_html_new(
        STATEMENT *statement,
        LIST *statement_prior_year_list(),
        double debit_sum,
        double credit_sum );

/* Process */
TRIAL_BALANCE_SUBCLASS_OMIT_HTML *
    trial_balance_subclass_omit_html_calloc(
        void );

LIST *trial_balance_subclass_omit_html_column_list(
    statement_prior_year_list );
```

```
LIST *row_list = list_new();

for ELEMENT *element in element_statement_list
{
    if ( !element->sum ) continue;

    LIST *trial_balance_subclass_omit_html_row_list(
        statement->end_date_time_string,
        element->element_name,
        element->accumulate_debit,
        element->account_statement_list,
        statement_prior_year_list );

    list_set_list(
        row_list,
        trial_balance_subclass_omit_html_row_list() );
}

HTML_ROW *trial_balance_html_total_row(
    debit_sum,
    credit_sum,
    2 /* column_skip_count */ );

list_set(
    row_list,
    trial_balance_html_total_row() );

/* Attributes */
LIST *column_list;
LIST *row_list;
```



TRIAL_BALANCE_SUBCLASS_OMIT_HTML (2)

```
/* Usage */
LIST *trial_balance_subclass_omit_html_column_list(
    LIST *statement_prior_year_list );

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

list_set(
    column_list,
    html_column_new(
       STATEMENT_ELEMENT_HEADING,
       0 /* not right_justify_boolean */ ) );

list_set(
    column_list,
    html_column_new(
       STATEMENT_ACCOUNT_HEADING,
       0 /* not right_justify_boolean */ ) );

list_set(
    column_list,
    html_column_new(
       STATEMENT_COUNT_HEADING,
       1 /* right_justify_boolean */ ) );

list_set(
    column_list,
    html_column_new(
       STATEMENT_DEBIT_HEADING,
       1 /* right_justify_boolean */ ) );

list_set(
    column_list,
    html_column_new(
       STATEMENT_CREDIT_HEADING,
       1 /* right_justify_boolean */ ) );

list_set(
    column_list,
    html_column_new(
       STATEMENT_PERCENT_OF_ASSET_HEADING,
       1 /* right_justify_boolean */ ) );

list_set(
    column_list,
    html_column_new(
       STATEMENT_PERCENT_OF_INCOME_HEADING,
       1 /* right_justify_boolean */ ) );

if ( list_length( statement_prior_year_list ) )
{
    LIST *statement_prior_year_heading_list(
       statement_prior_year_list );

    LIST *html_column_right_list(
       statement_prior_year_heading_list() );

    list_set_list(
       column_list,
       html_column_right_list() );
}
```

TRIAL_BALANCE_SUBCLASS_OMIT_HTML (3)

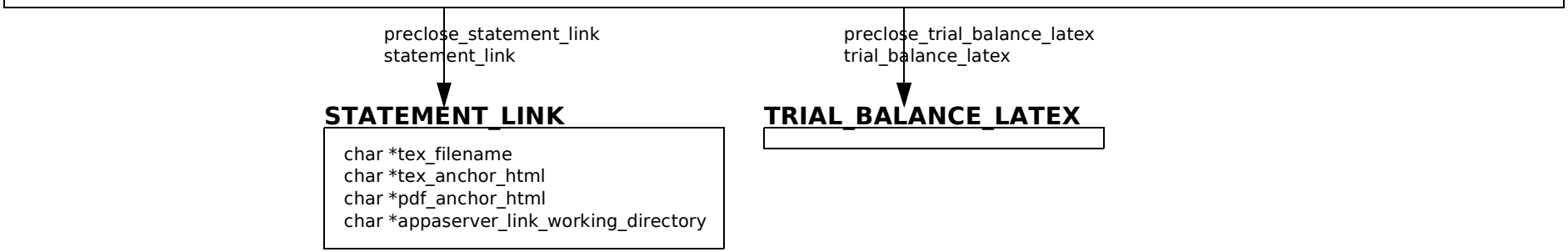
```
/* Usage */
LIST *trial_balance_subclass_omit_html_row_list(
    char *end_date_time_string,
    char *element_name,
    boolean element_accumulate_debit,
    LIST *account_statement_list,
    LIST *statement_prior_year_list );

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

for ACCOUNT *account in account_statement_list
{
    STATEMENT_ACCOUNT *statement_account =
        statement_account_new(
            end_date_time_string,
            element_accumulate_debit,
            account->account_journal_latest,
            account->action_string,
            0 /* not round_dollar_boolean */,
            account );

    HTML_ROW *trial_balance_subclass_omit_html_row(
        element_name,
        statement_account,
        statement_prior_year_list ) );

    list_set(
        row_list,
        trial_balance_subclass_omit_html_row() );

    element_name = (char *)0;
}
```

TRIAL_BALANCE_SUBCLASS_OMIT_HTML (4)

```
/* Usage */
HTML_ROW *trial_balance_subclass_omit_html_row(
    char *element_name,
    STATEMENT_ACCOUNT *statement_account,
    LIST *statement_prior_year_list );

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

char *statement_cell_label_datum(
    element_name );

list_set(
    cell_list,
    html_cell_new(
        statement_cell_label_datum(),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

char *transaction_date_american =
    char *statement_date_american(
        statement_account->
            account->
            account_journal_latest->
            transaction_date_time );

char *trial_balance_html_account_datum(
    statement_account->
        account->
        account_name,
    statement_account->
        account->
        account_journal_latest->
        full_name,
    statement_account->
        account->
        account_journal_latest->
        debit_amount,
    statement_account->
        account->
        account_journal_latest->
        credit_amount,
    transaction_date_american,
    statement_account->
        account->
        account_journal_latest->
        transaction->memo );

list_set(
    cell_list,
    html_cell_new(
        trial_balance_html_account_datum(),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

char *trial_balance_transaction_count_string(
    statement_account->
        account->
        transaction_count );

list_set(
    cell_list,
    html_cell_new(
        trial_balance_transaction_count_string(),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        statement_account->debit_string,
        0 /* not large_boolean */,
        statement_account->
            within_days_between_boolean
            /* bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        statement_account->credit_string,
        0 /* not large_boolean */,
        statement_account->
            within_days_between_boolean
            /* bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        statement_account->asset_percent_string,
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        statement_account->revenue_percent_string,
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

if ( list_length( statement_prior_year_list ) )
{
    LIST *statement_prior_year_account_data_list(
        statement_account->account->account_name,
        statement_prior_year_list ) );

    LIST *html_cell_list(
        statement_prior_year_account_data_list() );

    list_set(
        cell_list,
        html_cell_list() );
}

HTML_ROW *html_row_new( cell_list );
```

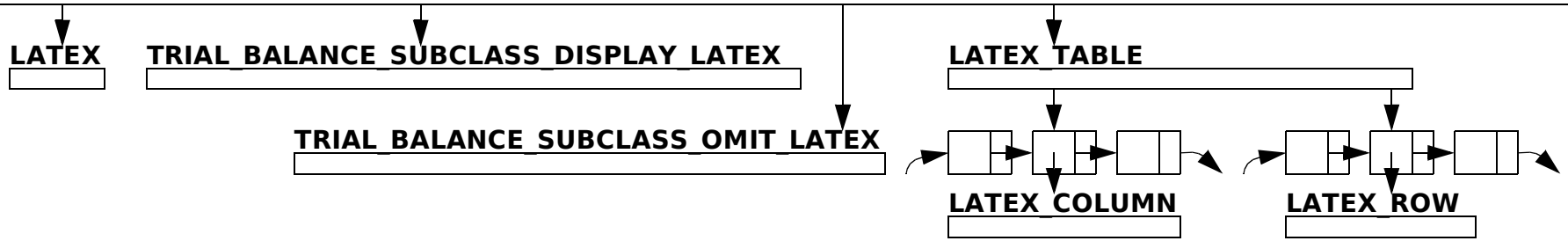
TRIAL_BALANCE_PDF

```
/* Usage */
TRIAL_BALANCE_PDF *trial_balance_pdf_new(
    char *application_name,
    char *appaserver_parameter_data_directory,
    enum statement_subclassification_option
      statement_subclassification_option,
    STATEMENT *preclose_statement,
    LIST *preclose_statement_prior_year_list,
    double preclose_debit_sum,
    double preclose_credit_sum,
    STATEMENT *statement,
    LIST *statement_prior_year_list,
    double debit_sum,
    double credit_sum,
    pid_t process_id );

/* Process */
TRIAL_BALANCE_PDF *trial_balance_pdf_calloc( void );

boolean statement_pdf_landscape_boolean(
    int list_length( statement_prior_year_list )
      /* statement_prior_year_list_length */ );

if ( preclose_statement )
{
    STATEMENT_LINK *preclose_statement_link =
      statement_link_new(
        application_name,
        preclose_statement->process_name,
        appaserver_parameter_data_directory,
        preclose_statement->
          transaction_date_begin_date_string,
        preclose_statement->
          end_date_time_string,
        process_id );

    TRIAL_BALANCE_LATEX *
      preclose_trial_balance_latex =
        trial_balance_latex_new(
          statement_subclassification_option,
          preclose_statement_link->
            tex_filename,
          preclose_statement_link->
            appaserver_link_working_directory,
          statement_pdf_landscape_boolean(),
          preclose_statement->
            statement_caption->
            logo_filename,
          preclose_statement,
          preclose_statement_prior_year_list,
          preclose_debit_sum,
          preclose_credit_sum );
}

STATEMENT_LINK *statement_link =
    statement_link_new(
        application_name,
        statement->process_name,
        appaserver_parameter_data_directory,
        statement->transaction_date_begin_date_string,
        statement->end_date_time_string,
        process_id );

TRIAL_BALANCE_LATEX *trial_balance_latex =
    trial_balance_latex_new(
        statement_subclassification_option,
        statement_link->tex_filename,
        statement_link->
          appaserver_link_working_directory,
        statement_pdf_landscape_boolean(),
        statement->
          statement_caption->
          logo_filename,
        statement,
        statement_prior_year_list,
        debit_sum,
        credit_sum );

/* Attributes */
boolean statement_pdf_landscape_boolean;
STATEMENT_LINK *preclose_statement_link;
TRIAL_BALANCE_LATEX *preclose_trial_balance_latex;
STATEMENT_LINK *statement_link;
TRIAL_BALANCE_LATEX *trial_balance_latex;
```



TRIAL_BALANCE_LATEX (1)

```
/* Usage */
TRIAL_BALANCE_LATEX *trial_balance_latex_new(
    enum statement_subclassification_option
       statement_subclassification_option,
    char *statement_link->tex_filename,
    char *statement_link->
       appaserver_link_working_directory,
    boolean statement_pdf_landscape_boolean(),
    char *statement_caption_logo_filename(),
    STATEMENT *statement,
    LIST *statement_prior_year_list,
    double debit_sum,
    double credit_sum );

/* Process */
TRIAL_BALANCE_LATEX *trial_balance_latex_calloc(
    void );

LATEX *latex =
    latex_new(
       tex_filename,
       appaserver_link_working_directory,
       statement_pdf_landscape_boolean,
       statement_caption_logo_filename );

if (statement_subclassification_option ==
    subclassification_display )
{
    TRIAL_BALANCE_SUBCLASS_DISPLAY_LATEX *
       trial_balance_subclass_display_latex =
          trial_balance_subclass_display_latex_new(
             statement,
             statement_prior_year_list,
             debit_sum,
             credit_sum );

    LATEX_TABLE *latex_table =
       LATEX_TABLE *latex_table_new(
             statement->statement_caption->combined,
             trial_balance_subclass_display_latex->
                column_list,
             trial_balance_subclass_display_latex->
                row_list );
}
else
{
    TRIAL_BALANCE_SUBCLASS_OMIT_LATEX *
       trial_balance_subclass_omit_latex =
          trial_balance_subclass_omit_latex_new(
             statement,
             statement_prior_year_list,
             debit_sum,
             credit_sum );

    LATEX_TABLE *latex_table =
       LATEX_TABLE *latex_table_new(
             statement->statement_caption->combined,
             trial_balance_subclass_omit_latex->
                column_list,
             trial_balance_subclass_omit_latex->
                row_list );
}

/* Usage */
char *trial_balance_latex_account_datum(
    char *account->account_name,
    char *account->
       account_journal_latest->
       full_name,
    double account->
       account_journal_latest->
       debit_amount,
    double account->
       account_journal_latest->
       credit_amount,
    char *statement_date_american(
       account->
          account_journal_latest->
          transaction_date_time )
             /* transaction_date_american */,
    char *account->
       account_journal_latest->
       memo );

/* Process */

/* Attributes */
LATEX *latex;
TRIAL_BALANCE_SUBCLASS_DISPLAY_LATEX *
    trial_balance_subclass_display_latex;
TRIAL_BALANCE_SUBCLASS_OMIT_LATEX *
    trial_balance_subclass_omit_latex;
LATEX_TABLE *latex_table;
```

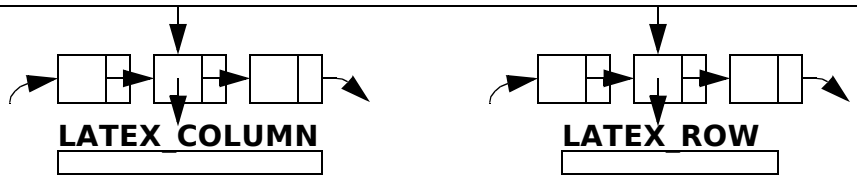


TRIAL_BALANCE_LATEX (2)

```
/* Usage */
LATEX_ROW *trial_balance_latex_total_row(
    double debit_sum,
    double credit_sum,
    LIST *latex_column_list,
    int column_skip_count );

/* Process */
list_rewind( latex_column_list );

LIST *cell_list = list_new();

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        “Total” /* datum */ ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

for ( i = 0; i < column_skip_count; i++ )
{
    LATEX_COLUMN *latex_column =
        list_get( latex_column_list );

    list_set(
        cell_list,
        latex_cell_small_new(
            latex_column,
            0 /* not first_row_boolean */,
            (char *)0 /* datum */ ) );

    LATEX_COLUMN *latex_column =
        list_get( latex_column_list );

    list_next( latex_column_list );
}

char *string_commas_double(
    debit_sum,
    0 /* decimal_place_count */ );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        strdup( string_commas_double() ) /* datum */ ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

char *string_commas_double(
    credit_sum,
    0 /* decimal_place_count */ );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        strdup( string_commas_double() ) /* datum */ ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

while ( list_still_more( latex_column_list ) )
{
    list_set(
        cell_list,
        latex_cell_small_new(
            latex_column,
            0 /* not first_row_boolean */,
            (char *)0 /* datum */ ) );

    LATEX_COLUMN *latex_column =
        list_get( latex_column_list );

    list_next( latex_column_list );
}

LATEX_ROW *latex_row_new( cell_list );
```

TRIAL_BALANCE_SUBCLASS_DISPLAY_LATEX (1)

```
/* Usage */
TRIAL_BALANCE_SUBCLASS_DISPLAY_LATEX *
    trial_balance_subclass_display_latex_new(
        STATEMENT *statement,
        LIST *statement_prior_year_list(),
        double debit_sum,
        double credit_sum );

/* Process */
TRIAL_BALANCE_SUBCLASS_DISPLAY_LATEX *
    trial_balance_subclass_display_latex_calloc(
        void );

LIST *trial_balance_subclass_display_latex_column_list(
    statement_prior_year_list );

LIST *row_list = list_new();

for ELEMENT *element in
    statement->element_statement_list
{
    if ( !element->sum ) continue;

    char *element_name = element->element_name;

    for SUBCLASSIFICATION *subclassification in
        element->subclassification_statement_list
    {
        if ( !subclassification->sum ) continue;

        LIST *trial_balance_subclass_display_latex_row_list(
            statement->transaction_date_time_closing,
            element_name,
            element->accumulate_debit,
            subclassification->subclassification_name,
            subclassification->account_statement_list,
            statement_prior_year_list,
            trial_balance_..._latex_column_list() );

        list_set_list(
            row_list,
            trial_balance_subclass_display_latex_row_list() );

        element_name = (char *)0;
    }
}

LATEX_ROW *trial_balance_latex_total_row(
    debit_sum,
    credit_sum,
    latex_column_list,
    3 /* column_skip_count */ );

list_set(
    row_list,
    trial_balance_latex_total_row() );

/* Attributes */
LIST *column_list;
LIST *row_list;
```



TRIAL_BALANCE_SUBCLASS_DISPLAY_LATEX (2)

```
/* Usage */
LIST *trial_balance_subclass_display_latex_column_list(
    LIST *statement_prior_year_list );

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

list_set(
    column_list,
    latex_column_new(
        “element” /* heading_string */,
        latex_column_text /* latex_column_enum */,
        0 /* float_decimal_count */,
        (char *)0 /* paragraph_size */,
        1 /* first_column_boolean */ ) );

list_set(
    column_list,
    latex_column_new(
        “subclassification” /* heading_string */,
        latex_column_text /* latex_column_enum */,
        0 /* float_decimal_count */,
        (char *)0 /* paragraph_size */,
        0 /* not first_column_boolean */ ) );

list_set(
    column_list,
    latex_column_new(
        “account” /* heading_string */,
        latex_column_text /* latex_column_enum */,
        0 /* float_decimal_count */,
        “5.5cm” /* paragraph_size */,
        0 /* not first_column_boolean */ ) );

list_set(
    column_list,
    latex_column_new(
        “count” /* heading_string */,
        latex_column_integer /* latex_column_enum */,
        0 /* float_decimal_count */,
        (char *)0 /* paragraph_size */,
        0 /* not first_column_boolean */ ) );

list_set(
    column_list,
    latex_column_new(
        “debit” /* heading_string */,
        latex_column_float /* latex_column_enum */,
        0 /* float_decimal_count */,
        (char *)0 /* paragraph_size */,
        0 /* not first_column_boolean */ ) );

list_set(
    column_list,
    latex_column_new(
        “credit” /* heading_string */,
        latex_column_float /* latex_column_enum */,
        0 /* float_decimal_count */,
        (char *)0 /* paragraph_size */,
        0 /* not first_column_boolean */ ) );

LATEX_COLUMN *latex_column =
    latex_column_new(
        “standardized” /* heading_string */,
        latex_column_text /* latex_column_enum */,
        0 /* float_decimal_count */,
        (char *)0 /* paragraph_size */,
        0 /* not first_column_boolean */ ) );

latex_column->right_justify_boolean  = 1;

list_set(
    column_list,
    latex_column );

if ( list_length( statement_prior_year_list ) )
{
    LIST *statement_prior_year_heading_list(
        statement_prior_year_list );

    LIST *latex_column_text_list(
        statement_prior_year_heading_list(),
        0 /* not first_column_boolean */,
        1 /* right_justify_boolean */ );

    list_set_list(
        column_list,
        latex_column_text_list() );
}
```

TRIAL_BALANCE_SUBCLASS_DISPLAY_LATEX (3)

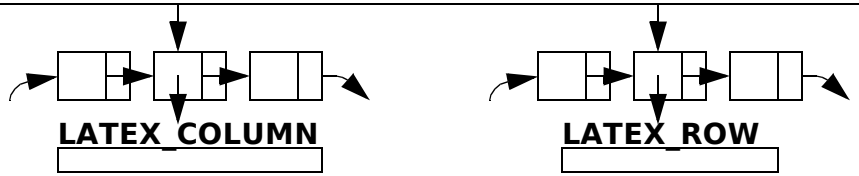
```
/* Usage */
LIST *trial_balance_subclass_display_latex_row_list(
    char *end_date_time_string,
    char *element_name,
    boolean element->accumulate_debit,
    char *subclassification->subclassification_name,
    LIST *subclassification->account_statement_list,
    LIST *statement_prior_year_list,
    LIST *latex_column_list );

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

for ACCOUNT *account in account_statement_list
{
    STATEMENT_ACCOUNT *statement_account =
        statement_account_new(
            end_date_time_string,
            accumulate_debit,
            account->account_journal_latest,
            (char *)0 /* account_action_string */,
            1 /* round_dollar_boolean */,
            account );

    LATEX_ROW *
    trial_balance_subclass_display_latex_row(
        element_name,
        subclassification_name,
        statement_account,
        statement_prior_year_list,
        latex_column_list );

    list_set(
        row_list,
        trial_balance_subclass_display_latex_row() );

    element_name = (char *)0;
    subclassification_name = (char *)0;
}
```

TRIAL_BALANCE_SUBCLASS_DISPLAY_LATEX (4)

```
/* Usage */
LATEX_ROW *trial_balance_subclass_display_latex_row(
    char *element_name,
    char *subclassifiction_name,
    STATEMENT_ACCOUNT *statement_account,
    LIST *statement_prior_year_list,
    LIST *latex_column_list );

/* Process */
list_rewind( latex_column_list );

LIST *cell_list = list_new();

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
list_next( latex_column_list );

char *statement_cell_label_datum( element_name );

LATEX_CELL *latex_cell_small_new(
    latex_column,
    0 /* not first_row_boolean */,
    statement_cell_label_datum() );

list_set( cell_list, latex_cell_small_new() );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
list_next( latex_column_list );

char *statement_cell_label_datum(
    subclassification_name );

LATEX_CELL *latex_cell_small_new(
    latex_column,
    0 /* not first_row_boolean */,
    statement_cell_label_datum() );

list_set( cell_list, latex_cell_small_new() );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
list_next( latex_column_list );

char *statement_date_american(
    statement_account->
      account->
      account_journal_latest->
      transaction_date_time );

char *trial_balance_latex_account_datum(
    statement_account->account->account_name,
    statement_account->
      account->
      account_journal_latest->
      full_name,
    statement_account->
      account->
      account_journal_latest->
      debit_amount,
    statement_account->
      account->
      account_journal_latest->
      credit_amount,
    statement_date_american(),
    statement_account->
      account->
      account_journal_latest->
      transaction->
      memo );

LATEX_CELL *latex_cell_new(
    latex_column,
    0 /* not first_row_boolean */,
    trial_balance_latex_account_datum(),
    latex_column->large_boolean,
    latex_column->bold_boolean );

list_set( cell_list, latex_cell_small_new() );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
list_next( latex_column_list );

char *trial_balance_transaction_count_string(
    statement_account->account->transaction_count );

LATEX_CELL *latex_cell_small_new(
    latex_column,
    0 /* not first_row_boolean */,
    trial_balance_transaction_count_string() );

list_set( cell_list, latex_cell_small_new() );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
list_next( latex_column_list );

LATEX_CELL *latex_cell_new(
    latex_column,
    0 /* not first_row_boolean */,
    statement_account->debit_string /* datum */,
    0 /* not large_boolean */,
    statement_account->within_days_between_boolean
      /* bold_boolean */ );

list_set( cell_list, latex_cell_new() );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
list_next( latex_column_list );

LATEX_CELL *latex_cell_new(
    latex_column,
    0 /* not first_row_boolean */,
    statement_account->credit_string /* datum */,
    0 /* not large_boolean */,
    statement_account->within_days_between_boolean
      /* bold_boolean */ );

list_set( cell_list, latex_cell_new() );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
list_next( latex_column_list );

LATEX_CELL *latex_cell_small_new(
    latex_column,
    0 /* not first_row_boolean */,
    statement_account->percent_string /* datum */ );

list_set( cell_list, latex_cell_small_new() );

if ( list_length( statement_prior_year_list ) )
{
    LIST *statement_prior_year_account_data_list(
      statement_account->account->account_name,
      statement_prior_year_list );

    LIST *latex_cell_list(
      __FUNCTION__,
      latex_column_list,
      statement_prior_year_account_data_list() );

    list_set_list(
      cell_list,
      latex_cell_list() );
}

LATEX_ROW *latex_row_new( cell_list );
```

TRIAL_BALANCE_SUBCLASS_OMIT_LATEX (1)

```
/* Usage */
TRIAL_BALANCE_SUBCLASS_OMIT_LATEX *
    trial_balance_subclass_omit_latex_new(
        STATEMENT *statement,
        LIST *statement_prior_year_list(),
        double debit_sum,
        double credit_sum );

/* Process */
TRIAL_BALANCE_SUBCLASS_OMIT_LATEX *
    trial_balance_subclass_omit_latex_calloc(
        void );

LIST *trial_balance_subclass_omit_latex_column_list(
    statement_prior_year_list );

LIST *row_list = list_new();

for ELEMENT *element in
    statement->element_statement_list
{
    if ( !element->sum ) continue;

    LIST *trial_balance_subclass_omit_latex_row_list(
        statement->end_date_time_string,
        element->element_name,
        element->accumulate_debit,
        element->account_statement_list,
        statement_prior_year_list,
        trial_balance_subclass_omit_latex_column_list() );

    list_set_list(
        row_list,
        trial_balance_subclass_omit_latex_row_list() );
}

LATEX_ROW *trial_balance_latex_total_row(
    debit_sum,
    credit_sum,
    trial_balance_subclass_omit_latex_column_list(),
    2 /* column_skip_count */ );

list_set(
    row_list,
    trial_balance_latex_total_row() );

/* Attributes */
LIST *column_list;
LIST *row_list;
```



TRIAL_BALANCE_SUBCLASS_OMIT_LATEX (2)

```
/* Usage */
LIST *trial_balance_subclass_omit_latex_column_list(
    LIST *statement_prior_year_list );

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

list_set(
    latex_column_list,
    latex_column_new(
        "element" /* heading_string */,
        latex_column_text /* latex_column_enum */,
        0 /* float_decimal_count */,
        (char *)0 /* paragraph_size */,
        1 /* first_column_boolean */ ) );

list_set(
    latex_column_list,
    latex_column_new(
        "account" /* heading_string */,
        latex_column_text /* latex_column_enum */,
        0 /* float_decimal_count */,
        "8.0cm" /* paragraph_size */,
        0 /* not first_column_boolean */ ) );

list_set(
    latex_column_list,
    latex_column_new(
        "count" /* heading_string */,
        latex_column_integer /* latex_column_enum */,
        0 /* float_decimal_count */,
        (char *)0 /* paragraph_size */,
        0 /* not first_column_boolean */ ) );

list_set(
    latex_column_list,
    latex_column_new(
        "debit" /* heading_string */,
        latex_column_float /* latex_column_enum */,
        0 /* float_decimal_count */,
        (char *)0 /* paragraph_size */,
        0 /* not first_column_boolean */ ) );

list_set(
    latex_column_list,
    latex_column_new(
        "credit" /* heading_string */,
        latex_column_float /* latex_column_enum */,
        0 /* float_decimal_count */,
        (char *)0 /* paragraph_size */,
        0 /* not first_column_boolean */ ) );

LATEX_COLUMN *latex_column =
    latex_column_new(
        "standardized" /* heading_string */,
        latex_column_text /* latex_column_enum */,
        0 /* float_decimal_count */,
        (char *)0 /* paragraph_size */,
        0 /* not first_column_boolean */ ) );

latex_column->right_justify_boolean = 1;

list_set(
    latex_column_list,
    latex_column );

if ( list_length( statement_prior_year_list ) )
{
    LIST *statement_prior_year_heading_list(
        statement_prior_year_list );

    LIST *latex_column_text_list(
        statement_prior_year_heading_list(),
        0 /* not first_column_boolean */,
        1 /* right_justify_boolean */ );

    list_set_list(
        latex_column_list,
        latex_column_text_list() );
}
```

TRIAL_BALANCE_SUBCLASS_OMIT_LATEX (3)

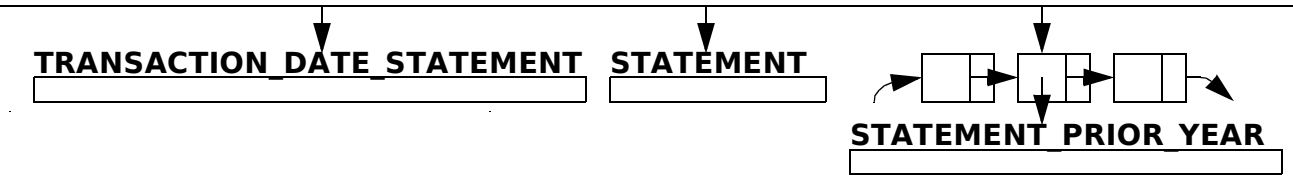
```
/* Usage */
LIST *trial_balance_subclass_omit_latex_row_list(
    char *statement->end_date_time_string,
    char *element->element_name,
    boolean element->accumulate_debit,
    LIST *element->account_statement_list,
    LIST *statement_prior_year_list(),
    LIST *latex_column_list );

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

for ACCOUNT *account in account_statement_list
{
    STATEMENT_ACCOUNT *statement_account =
      statement_account_new(
        end_date_time_string,
        element_accumulate_debit,
        account->account_journal_latest,
        (char *)0 /* account_action_string */,
        1 /* round_dollar_boolean */,
        account );

    LATEX_ROW *trial_balance_subclass_omit_latex_row(
        element_name,
        statement_account,
        statement_prior_year_list,
        latex_column_list );

    list_set(
        latex_row_list,
        trial_balance_subclass_omit_latex_row() );

    element_name = (char *)0;
}
```

TRIAL_BALANCE_SUBCLASS_OMIT_LATEX (4)

```
/* Usage */
LATEX_ROW *trial_balance_subclass_omit_latex_row(
    char *element_name,
    STATEMENT_ACCOUNT *statement_account,
    LIST *statement_prior_year_list,
    LIST *latex_column_list );

/* Process */
list_rewind( latex_column_list );

LIST *cell_list = list_new();

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

char *statement_cell_label_datum( element_name );

LATEX_CELL *latex_cell_small_new(
    latex_column,
    0 /* not first_row_boolean */,
    statement_cell_label_datum() );

list_set( cell_list, latex_cell_small_new() );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

char *statement_date_american(
    statement_account->
      account->
      account_journal_latest->
      transaction_date_time );

char *trial_balance_latex_account_datum(
    statement_account->account->account_name,
    statement_account->
      account->
      account_journal_latest->
      full_name,
    statement_account->
      account->
      account_journal_latest->
      debit_amount,
    statement_account->
      account->
      account_journal_latest->
      credit_amount,
    statement_date_american(),
    statement_account->
      account->
      account_journal_latest->
      transaction->
      memo );

LATEX_CELL *latex_cell_small_new(
    latex_column,
    0 /* not first_row_boolean */,
    trial_balance_latex_account_datum() );

list_set( cell_list, latex_cell_small_new() );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

char *trial_balance_transaction_count_string(
    statement_account->account->transaction_count );

LATEX_CELL *latex_cell_small_new(
    latex_column,
    0 /* not first_row_boolean */,
    trial_balance_transaction_count_string() );

list_set( cell_list, latex_cell_small_new() );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

LATEX_CELL *latex_cell_new(
    latex_column,
    0 /* not first_row_boolean */,
    statement_account->debit_string /* datum */,
    0 /* not large_boolean */,
    statement_account->within_days_between_boolean
      /* bold_boolean */ );

list_set( cell_list, latex_cell_new() );

list_next( latex_column_list );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

LATEX_CELL *latex_cell_new(
    latex_column,
    0 /* not first_row_boolean */,
    statement_account->credit_string /* datum */,
    0 /* not large_boolean */,
    statement_account->within_days_between_boolean
      /* bold_boolean */ );

list_set( cell_list, latex_cell_new() );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

LATEX_CELL *latex_cell_small_new(
    latex_column,
    0 /* not first_row_boolean */,
    statement_account->percent_string /* datum */ );

list_set( cell_list, latex_cell_small_new() );

if ( list_length( statement_prior_year_list ) )
{
    LIST *statement_prior_year_account_data_list(
      statement_account->account->account_name,
      statement_prior_year_list );

    LIST *latex_cell_list(
      __FUNCTION__,
      latex_column_list,
      statement_prior_year_account_data_list() );

    list_set_list(
      cell_list,
      latex_cell_list() );
}

LATEX_ROW *latex_row_new( cell_list );
```

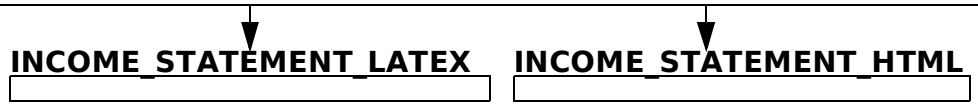
INCOME_STATEMENT (1)

```
/* Usage */
INCOME_STATEMENT *income_statement_fetch(
    char *argv_0,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *process_name,
    char *appaserver_parameter_data_directory,
    char *as_of_date_string,
    int prior_year_count,
    char *subclassification_option_string,
    char *output_medium_string );

/* Process */
INCOME_STATEMENT *income_statement_calloc( void );

enum statement_subclassification_option =
    statement_resolve_subclassification_option(
        subclassification_option_string );

enum statement_output_medium =
    statement_resolve_output_medium(
        output_medium_string );

TRANSACTION_DATE_STATEMENT *
    transaction_date_statement =
        transaction_date_statement_new(
            as_of_date_string );

if ( !transaction_date_statement->
        transaction_date_as_of )
{
    return NULL;
}

LIST *income_statement_element_name_list(
    char *ELEMENT_REVENUE,
    char *ELEMENT_EXPENSE,
    char *ELEMENT_GAIN,
    char *ELEMENT_LOSS );

STATEMENT *statement =
    statement_fetch(
        application_name,
        process_name,
        prior_year_count,
        income_statement_element_name_list(),
        transaction_date_statement->
            transaction_date_begin_date_string,
        transaction_date_statement->
            end_date_time_string,
        0 /* not fetch_transaction */ );

if ( statement_output_medium == output_table )
{
    void element_account_transaction_count_set(
        statement->element_statement_list,
        transaction_date_statement->
            transaction_date_begin_date_string,
        transaction_date_statement->
            end_date_time_string );

    void element_account_action_string_set(
        statement->element_statement_list,
        application_name,
        session_key,
        login_name,
        role_name,
        transaction_date_statement->
            transaction_date_begin_date_string,
        transaction_date_statement->
            end_date_time_string );
}

if ( prior_year_count )
{
    LIST *statement_prior_year_list(
        income_statement_element_name_list(),
        transaction_date_statement->
            end_date_time_string,
        prior_year_count,
        statement );
}
```



INCOME_STATEMENT (2)

```
/* Process (continued) */
if (statement_subclassification_option ==
    statement_subclassification_omit )
{
    element_list_account_statement_list_set(
       statement->
          element_statement_list );
}

double element_net_income(
    statement->element_statement_list );

char *income_statement_net_income_percent_of_income_display(
    const char *ELEMENT_REVENUE,
    const char *ELEMENT_GAIN,
    double element_net_income(),
    LIST *statement->element_statement_list );

char *income_statement_net_income_label(
    char *argv_0 );

if ( statement_output_medium == statement_output_PDF )
{
    INCOME_STATEMENT_LATEX *
       income_statement_latex =
          income_statement_latex_new(
             application_name,
             process_name,
             appaserver_parameter_data_directory,
             statement_subclassification_option,
             statement,
             statement_prior_year_list,
             element_net_income(),
             income_statement_net_income_percent_of_income_display(),
             income_statement_net_income_label(),
             getpid() /* process_id */ );
}
else
if ( statement_output_medium == statement_output_table )
{
    INCOME_STATEMENT_HTML *
       income_statement_html =
          income_statement_html_new(
             statement_subclassification_option,
             statement,
             statement_prior_year_list,
             element_net_income(),
             income_statement_net_income_percent_of_income_display(),
             income_statement_net_income_label() );
}
```

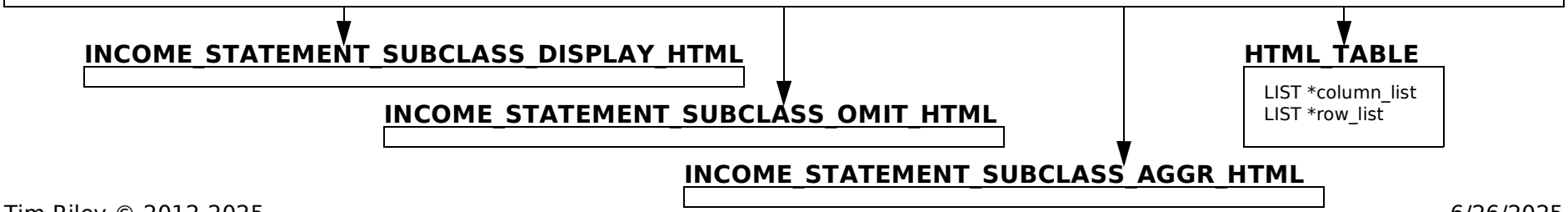


INCOME_STATEMENT (3)

```
/* Usage */
LIST *income_statement_prior_net_income_data_list(
    double current_net_income,
    LIST *statement_prior_year_list );

/* Process */
for STATEMENT_PRIOR_YEAR *statement_prior_year in
    statement_prior_year_list
{
    double prior_net_income =
       element_net_income(
          statement_prior_year->
             element_statement_list );

    int delta_prior_percent =
        float_delta_prior_percent(
           prior_net_income,
           current_net_income );

    list_set(
       data_list,
       char *statement_prior_year_cell_display(
        0 /* not cell_empty */,
        delta_prior_percent,
        prior_net_income /* prior_year_amount */ ) );
}

/* Usage */
double income_statement_fetch_net_income(
    char *end_date_time_string );

/* Process */
LIST *income_statement_element_name_list(
    ELEMENT_REVENUE,
    ELEMENT_EXPENSE,
    ELEMENT_GAIN,
    ELEMENT_LOSS );

STATEMENT *statement =
    statement_fetch(
       (char *)0 /* application_name */,
       (char *)0 /* process_name */,
       0 /* prior_year_count */,
       income_statement_element_name_list(),
       (char *)0 /* transaction_begin_date_string */,
       end_date_time_string,
       0 /* not fetch_transaction */ );

double element_net_income(
    statement->element_statement_list );

/* Driver */
printf(
    "%s\n",
    income_statement->
       statement->
       statement_caption->
       frame_title );

if ( income_statement_latex )
{
    void latex_table_output(
       income_statement->
          income_statement_latex->
          statement_link->
          tex_filename,
       income_statement_latex->
          statement_link->
          appaserver_link_working_directory,
       income_statement->
          income_statement_latex->
          statement_link->
          pdf_anchor_html,
       income_statement->
          income_statement_latex->
          latex,
       income_statement->
          income_statement_latex->
          latex_table );
}
else
if ( income_statement_html )
{
    void statement_html_output(
       income_statement_html->
          html_table,
          (char *)0 /* secondary_title */ );
}

/* Attributes */
enum statement_subclassification_option
    statement_subclassification_option;
enum statement_output_medium
    statement_output_medium;
TRANSACTION_DATE_STATEMENT *
    transaction_date_statement;
LIST *element_name_list;
STATEMENT *statement;
LIST *statement_prior_year_list;
double element_net_income;
char *net_income_percent_of_income_display;
char *net_income_label;
INCOME_STATEMENT_LATEX *income_statement_latex;
INCOME_STATEMENT_HTML *income_statement_html;
```

INCOME_STATEMENT_HTML (1)

```
/* Usage */
INCOME_STATEMENT_HTML *
    income_statement_html_new(
        enum statement_subclassification_option
            statement_subclassification_option,
        STATEMENT *statement,
        LIST *statement_prior_year_list,
        double element_net_income(),
        char *income_statement_net_income_percent_of_income_display(),
        char *income_statement_net_income_label() );

/* Process */
INCOME_STATEMENT_HTML *income_statement_html_calloc( void );

if (statement_subclassification_option == statement_subclassification_display )
{
    INCOME_STATEMENT_SUBCLASS_DISPLAY_HTML *
        income_statement_subclass_display_html =
            income_statement_subclass_display_html_new(
                statement,
                statement_prior_year_list,
                element_net_income,
                income_statement_net_income_percent_of_income_display,
                income_statement_net_income_label );

    HTML_TABLE *income_statement_html_table(
        statement->caption_sub_title,
        income_statement_subclass_display_html->
            statement_subclass_display_html_list->
            column_list,
        income_statement_subclass_display_html->
            row_list );
}
else
if ( statement_subclassification_option == statement_subclassification_omit )
{
    INCOME_STATEMENT_SUBCLASS_OMIT_HTML *
        income_statement_subclass_omit_html =
            income_statement_subclass_omit_html_new(
                statement,
                statement_prior_year_list,
                element_net_income,
                income_statement_net_income_percent_of_income_display,
                income_statement_net_income_label );

    HTML_TABLE *income_statement_html_table(
        statement->caption_sub_title,
        income_statement_subclass_omit_html->
            statement_subclass_omit_html_list->
            column_list,
        income_statement_subclass_omit_html->
            row_list );
}
else
if ( statement_subclassification_option == statement_subclassification_aggregate )
{
    INCOME_STATEMENT_SUBCLASS_AGGR_HTML *
        income_statement_subclass_aggr_html =
            income_statement_subclass_aggr_html_new(
                statement,
                statement_prior_year_list,
                element_net_income,
                income_statement_net_income_percent_of_income_display,
                income_statement_net_income_label );

    HTML_TABLE *income_statement_html_table(
        statement->caption_sub_title,
        income_statement_subclass_aggr_html->
            statement_subclass_aggr_html_list->
                column_list,
        income_statement_subclass_aggr_html->
            row_list );
}

/* Attributes */
INCOME_STATEMENT_SUBCLASS_DISPLAY_HTML *
    income_statement_subclass_display_html;
INCOME_STATEMENT_SUBCLASS_OMIT_HTML *
    income_statement_subclass_omit_html;
INCOME_STATEMENT_SUBCLASS_AGGR_HTML *
    income_statement_subclass_aggr_html;
HTML_TABLE *html_table;
```

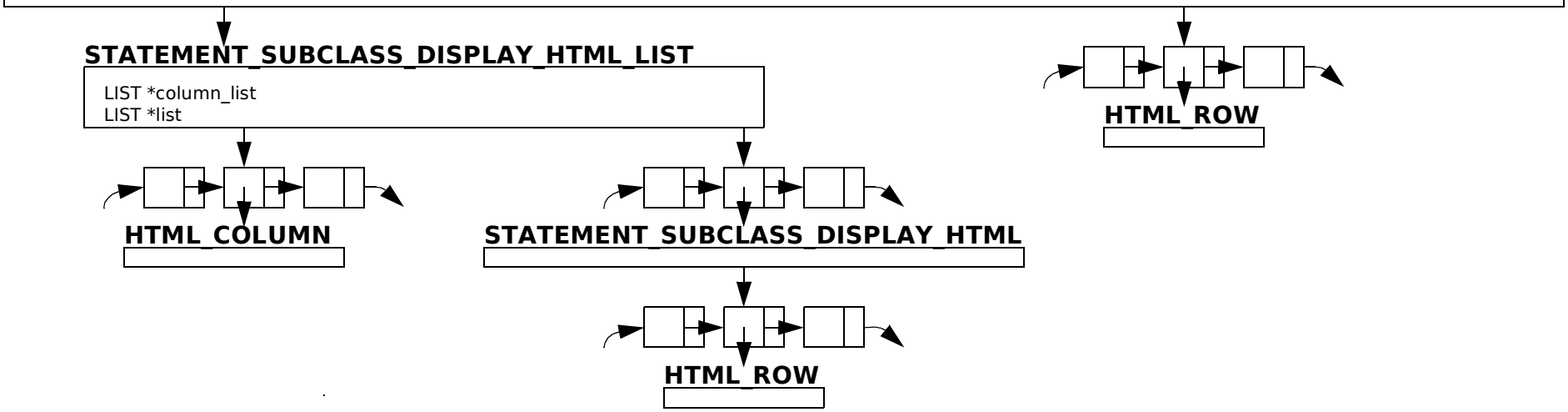


INCOME_STATEMENT_HTML (2)

```
/* Usage */
HTML_TABLE *income_statement_html_table(
    char *statement_caption_sub_title,
    LIST *column_list,
    LIST *row_list );

/* Process */
HTML_TABLE *html_table =
    html_table_new(
       (char *)0 /* title */,
       statement_caption_sub_title,
       (char *)0 /* sub_sub_title */ );

html_table->column_list = column_list;
html_table->row_list = row_list;

/* Usage */
HTML_ROW *income_statement_html_net_income_row(
    LIST *statement_prior_year_list,
    double element_net_income(),
    char *percent_of_income_display,
    char *net_income_label,
    int empty_cell_count );

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

char *statement_cell_label_datum(
    net_income_label );

list_set(
    cell_list,
    html_cell_new(
       statement_cell_label_datum(),
       1 /* large_boolean */,
       1 /* bold_boolean */ ) );

for ( i = 0;
       i < empty_cell_count;
       i++ )
{
    list_set(
       cell_list,
       html_cell_new(
          (char *)0 /* datum */,
          0 /* not large_boolean */,
          0 /* not bold_boolean */ ) );
}

char *string_commas_money( element_net_income );

list_set(
    cell_list,
    html_cell_new(
       strdup( string_commas_money() ),
       0 /* not large_boolean */,
       0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
       percent_of_income_display,
       0 /* not large_boolean */,
       0 /* not bold_boolean */ ) );

if ( list_length( statement_prior_year_list ) )
{
    LIST *income_statement_prior_net_income_data_list(
       element_net_income /* current_net_income */,
       statement_prior_year_list );

    LIST *html_cell_list(
       income_statement_prior_net_income_data_list() );

    list_set_list(
       cell_list,
       html_cell_list() );
}

HTML_ROW *html_row_new( cell_list );
```

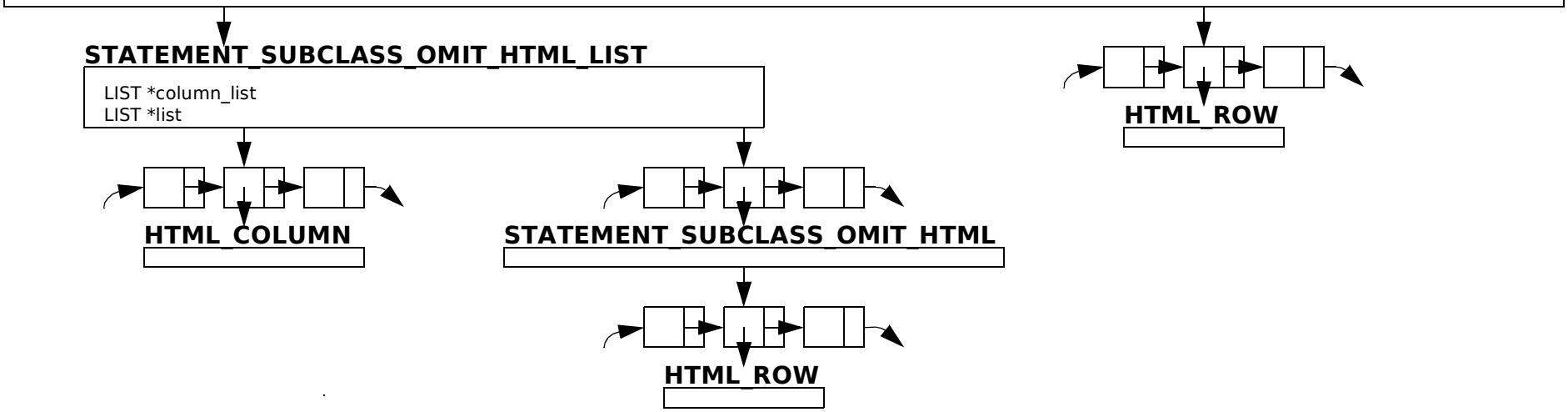
INCOME_STATEMENT_SUBCLASS_DISPLAY_HTML

```
/* Usage */
INCOME_STATEMENT_SUBCLASS_DISPLAY_HTML *
    income_statement_subclass_display_html_new(
        STATEMENT *statement,
        LIST *statement_prior_year_list(),
        double element_net_income(),
        char *income_statement_net_income_percent_of_income_display(),
        char *income_statement_net_income_label() );

/* Process */
INCOME_STATEMENT_SUBCLASS_DISPLAY_HTML *
    income_statement_subclass_display_html_calloc(
        void );

STATEMENT_SUBCLASS_DISPLAY_HTML_LIST *
    statement_subclass_display_html_list =
        statement_subclass_display_html_list_new(
            statement->element_statement_list,
            statement_prior_year_list );

LIST *row_list =
    statement_subclass_display_html_list_row_list(
        statement_subclass_display_html_list );

list_set(
    row_list,
    HTML_ROW *income_statement_html_net_income_row(
        statement_prior_year_list,
        element_net_income,
        income_statement_net_income_percent_of_income_display,
        income_statement_net_income_label,
        2 /* empty_cell_count */ ) );

/* Attributes */
STATEMENT_SUBCLASS_DISPLAY_HTML_LIST *
    statement_subclass_display_html_list;
LIST *row_list;
```



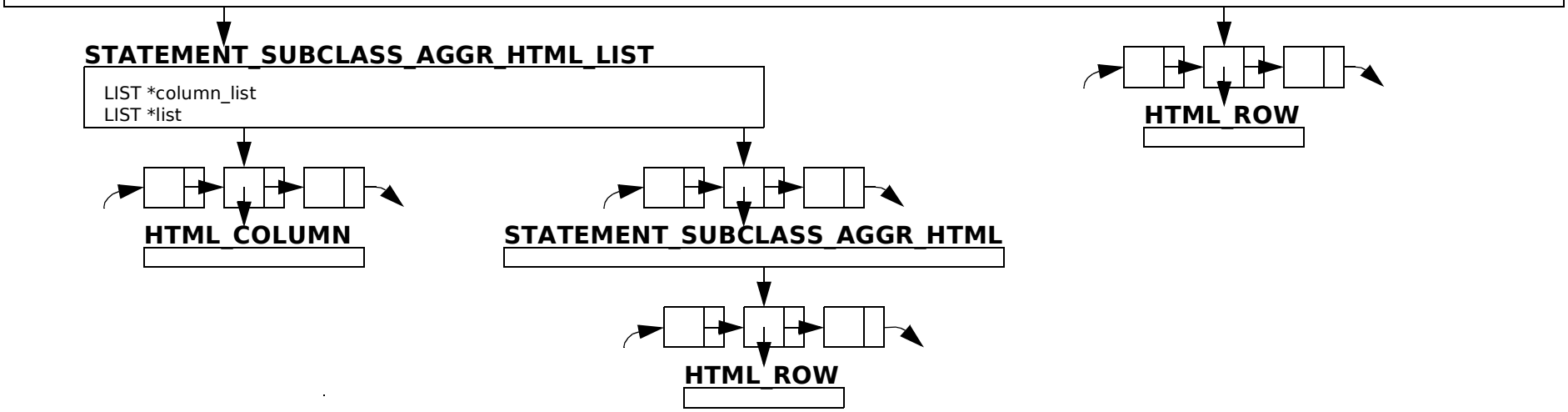
INCOME_STATEMENT_SUBCLASS_OMIT_HTML

```
/* Usage */
INCOME_STATEMENT_SUBCLASS_OMIT_HTML *
    income_statement_subclass_omit_html_new(
        STATEMENT *statement,
        LIST *statement_prior_year_list(),
        double element_net_income(),
        char *income_statement_net_income_percent_of_income_display(),
        char *income_statement_net_income_label() );

/* Process */
INCOME_STATEMENT_SUBCLASS_OMIT_HTML *
    income_statement_subclass_omit_html_calloc(
        void );

STATEMENT_SUBCLASS_OMIT_HTML_LIST *
    statement_subclass_omit_html_list =
        statement_subclass_omit_html_list_new(
            statement->element_statement_list,
            statement_prior_year_list );

LIST *row_list =
    statement_subclass_omit_html_list_row_list(
        statement_subclass_omit_html_list );

list_set(
    row_list,
    HTML_ROW *income_statement_html_net_income_row(
        statement_prior_year_list,
        element_net_income,
        income_statement_net_income_percent_of_income_display,
        income_statement_net_income_label,
        1 /* empty_cell_count */ ) );

/* Attributes */
STATEMENT_SUBCLASS_OMIT_HTML_LIST *
    statement_subclass_omit_html_list;
LIST *row_list;
```



INCOME_STATEMENT_SUBCLASS_AGGR_HTML

```
/* Usage */
INCOME_STATEMENT_SUBCLASS_AGGR_HTML *
    income_statement_subclass_aggr_html_new(
        STATEMENT *statement,
        LIST *statement_prior_year_list(),
        double element_net_income(),
        char *income_statement_net_income_percent_of_income_display(),
        char *income_statement_net_income_label() );

/* Process */
INCOME_STATEMENT_SUBCLASS_AGGR_HTML *
    income_statement_subclass_aggr_html_calloc(
        void );

STATEMENT_SUBCLASS_AGGR_HTML_LIST *
    statement_subclass_aggr_html_list =
        statement_subclass_aggr_html_list_new(
            statement->element_statement_list,
            statement_prior_year_list );

LIST *row_list =
    statement_subclass_aggr_html_list_row_list(
        statement_subclass_aggr_html_list );

list_set(
    row_list,
    HTML_ROW *income_statement_html_net_income_row(
        statement_prior_year_list,
        element_net_income,
        income_statement_net_income_percent_of_income_display,
        income_statement_net_income_label,
        1 /* empty_cell_count */ ) );

/* Attributes */
STATEMENT_SUBCLASS_AGGR_HTML_LIST *
    statement_subclass_aggr_html_list;
LIST *row_list;
```

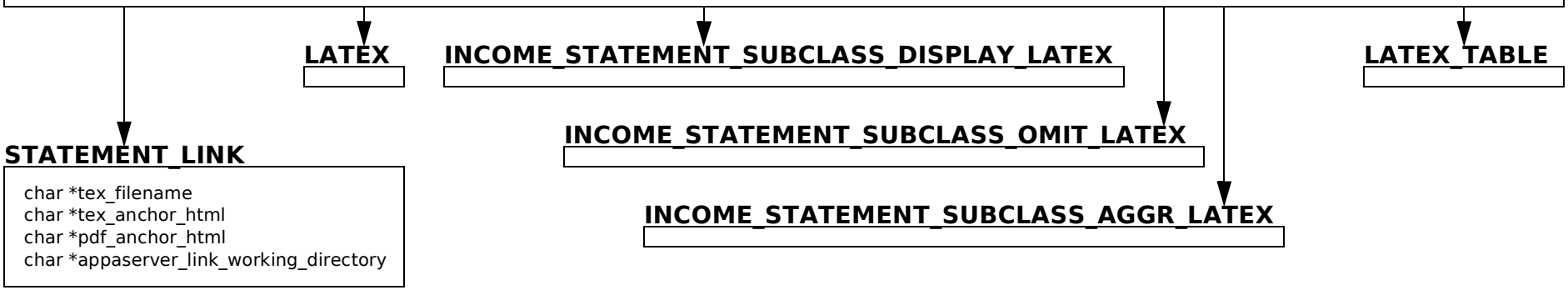


INCOME_STATEMENT_LATEX (1)

```
/* Usage */
INCOME_STATEMENT_LATEX *
    income_statement_latex_new(
       char *application_name,
       char *process_name,
       char *appaserver_parameter_data_directory,
       enum statement_subclassification_option
          statement_subclassification_option,
       STATEMENT *statement,
       LIST *statement_prior_year_list,
       double element_net_income(),
       char *income_statement_net_income_percent_of_income_display(),
       char *income_statement_net_income_label(),
       pid_t process_id );

/* Process */
INCOME_STATEMENT_LATEX *income_statement_latex_calloc(
    void );

STATEMENT_LINK *statement_link =
    statement_link_new(
       application_name,
       process_name,
       appaserver_parameter_data_directory,
       statement->transaction_date_begin_date_string,
       statement->end_date_time_string,
       process_id );

LATEX *latex = latex_new(
    statement_link->tex_filename,
    statement_link->appaserver_link_working_directory,
    statement->pdf_landscape_boolean,
    statement->statement_caption->logo_filename );

if (statement_subclassification_option == statement_subclassification_display )
{
    INCOME_STATEMENT_SUBCLASS_DISPLAY_LATEX *
       income_statement_subclass_display_latex =
          income_statement_subclass_display_latex_new(
             statement,
             statement_prior_year_list,
             element_net_income,
             net_income_percent_of_income_display,
             net_income_label );

    LATEX_TABLE *latex_table =
       LATEX_TABLE *income_statement_latex_table(
             statement->statement_caption->combined,
             income_statement_subclass_display_latex_list->column_list
                /* latex_column_list */,
             income_statement_subclass_display_latex->row_list
                /* latex_row_list */ );
}
else
if ( statement_subclassification_option == statement_subclassification_omit )
{
    INCOME_STATEMEMENT_SUBCLASS_OMIT_LATEX *
       income_statement_subclass_omit_latex =
          income_statement_subclass_omit_latex_new(
             statement,
             statement_prior_year_list,
             element_net_income,
             net_income_percent_of_income_display,
             net_income_label );

    LATEX_TABLE *latex_table =
       LATEX_TABLE *income_statement_latex_table(
             statement->statement_caption->combined,
             income_statement_subclass_omit_latex_list->column_list
                /* latex_column_list */,
             income_statement_subclass_omit_latex->row_list
                /* latex_row_list */ );
}
else
if ( statement_subclassification_option == statement_subclassification_aggregate )
{
    INCOME_STATEMEMENT_SUBCLASS_AGGR_LATEX *
       income_statement_subclass_aggr_latex =
          income_statement_subclass_aggr_latex_new(
             statement,
             statement_prior_year_list,
             element_net_income,
             net_income_percent_of_income_display,
             net_income_label );

    LATEX_TABLE *latex_table =
       LATEX_TABLE *income_statement_latex_table(
             statement->statement_caption->combined,
             income_statement_subclass_aggr_latex_list->column_list
                /* latex_column_list */,
             income_statement_subclass_aggr_latex->row_list
                /* latex_row_list */ );
}
```

INCOME_STATEMENT_LATEX (2)

```
/* Attributes */
STATEMENT_LINK *statement_link;
INCOME_STATEMENT_SUBCLASS_DISPLAY_LATEX *
    income_statement_subclass_display_latex;
INCOME_STATEMENT_SUBCLASS_OMIT_LATEX *
    income_statement_subclass_omit_latex;
INCOME_STATEMENT_SUBCLASS_AGGR_LATEX *
    income_statement_subclass_aggr_latex;
LATEX *latex;
LATEX_TABLE *latex_table;
```

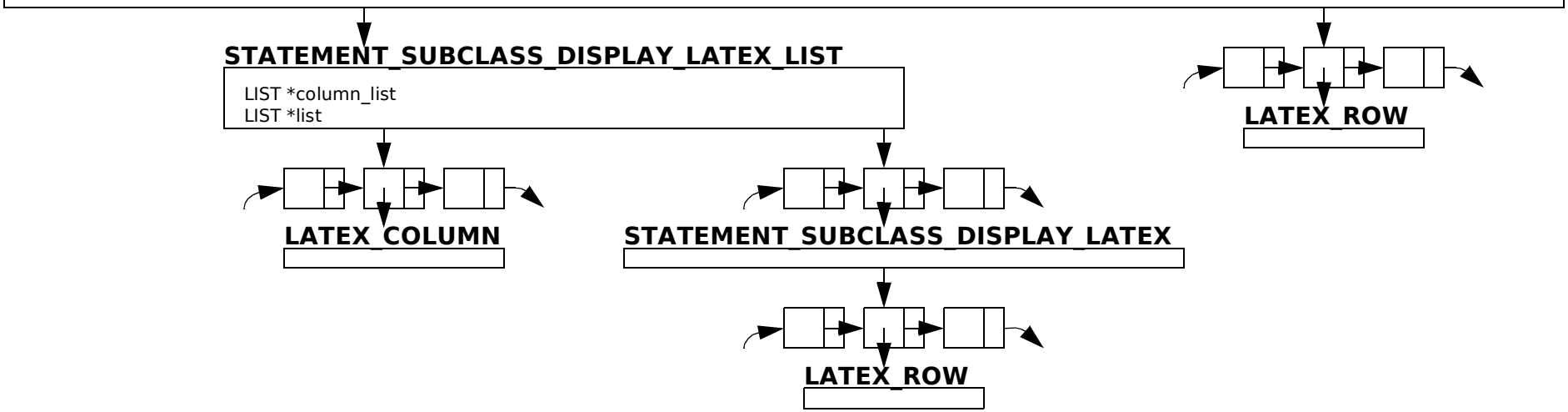


INCOME_STATEMENT_LATEX (3)

```
/* Usage */
LATEX_TABLE *income_statement_latex_table(
    char *statement_caption_combined,
    LIST *latex_column_list,
    LIST *latex_row_list );

/* Process */
char *date_now16( date_utc_offset() );

LATEX_TABLE *latex_table =
    latex_table_new(
        statement_caption_combined /* title */,
        latex_column_list,
        latex_row_list );

/* Usage */
LATEX_ROW *income_statement_latex_net_income_row(
    LIST *statement_prior_year_list,
    double element_net_income(),
    char *percent_of_income_display,
    char *net_income_label(),
    LIST *latex_column_list,
    int empty_cell_count );

/* Process */
list_rewind( latex_column_list );

LIST *cell_list = list_new();

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

char *statement_cell_label_datum(
    net_income_label ),

LATEX_CELL *latex_cell_new(
    latex_column,
    0 /* not first_row_boolean */,
    statement_cell_label_datum(),
    1 /* large_boolean */,
    1 /* bold_boolean */ );

list_set( cell_list, latex_cell_new() );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

for ( i = 0;
      i < empty_cell_count;
      i++ )
{
    list_set(
        cell_list,
        latex_cell_small_new(
            latex_column,
            (char *)0 /* datum */ ) );

    LATEX_COLUMN *latex_column =
        list_get( latex_column_list );

    list_next( latex_column_list );
}

char *string_commas_money( element_net_income );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        strdup( string_commas_money() ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        percent_of_income_display ) );

if ( list_length( statement_prior_year_list ) )
{
    LIST *income_statement_prior_net_income_data_list(
        element_net_income /* current_net_income */,
        statement_prior_year_list );

    LIST *latex_cell_list(
        __FUNCTION__,
        latex_column_list,
        statement_prior_year_net_income_data_list() );

    list_set_list(
        cell_list,
        latex_cell_list() );
}

LATEX_ROW *latex_row_new( cell_list );
```

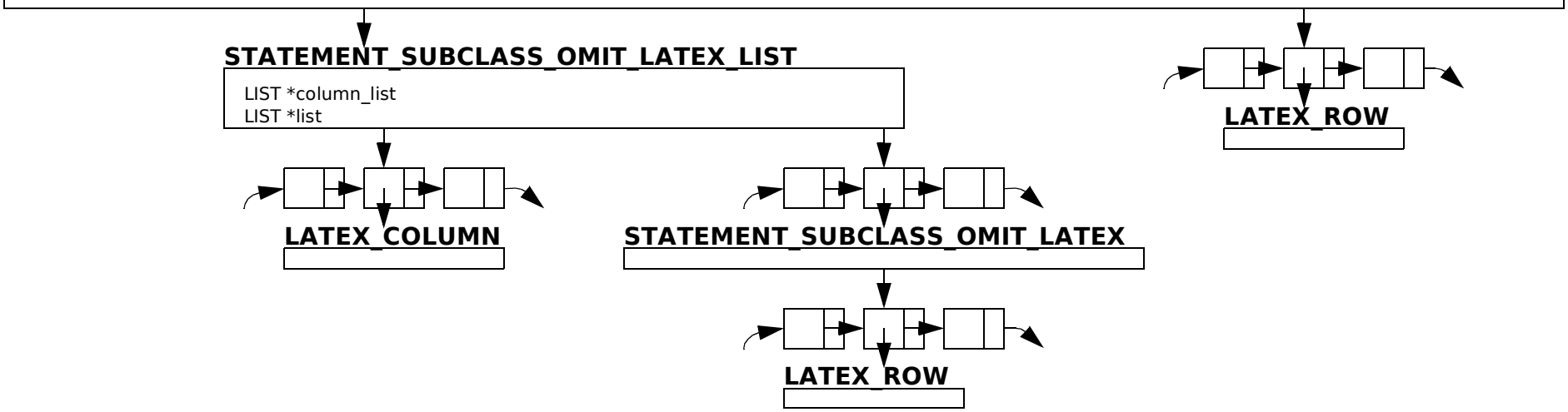
INCOME_STATEMENT_SUBCLASS_DISPLAY_LATEX

```
/* Usage */
INCOME_STATEMENT_SUBCLASS_DISPLAY_LATEX *
    income_statement_subclass_display_latex_new(
        STATEMENT *statement,
        LIST *statement_prior_year_list(),
        double element_net_income(),
        char *income_statement_net_income_percent_of_income_display(),
        char *income_statement_net_income_label() );

/* Process */
INCOME_STATEMENT_SUBCLASS_DISPLAY_LATEX *
    income_statement_subclass_display_latex_calloc(
        void );

STATEMENT_SUBCLASS_DISPLAY_LATEX_LIST *
    statement_subclass_display_latex_list =
        statement_subclass_display_latex_list_new(
            statement->element_statement_list,
            statement->statement_prior_year_list );

LIST *row_list =
    statement_subclass_display_latex_list_row_list(
        statement_subclass_display_latex_list );

list_set(
    row_list,
    LATEX_ROW *income_statement_latex_net_income_row(
        statement_prior_year_list,
        element_net_income,
        income_statement_net_income_percent_of_income_display,
        income_statement_net_income_label,
        statement_subclass_display_latex_list->column_list
            /* latex_column_list */,
        2 /* empty_cell_count */ );

/* Attributes */
STATEMENT_SUBCLASS_DISPLAY_LATEX_LIST *
    statement_subclass_display_latex_list;
LIST *row_list;
```



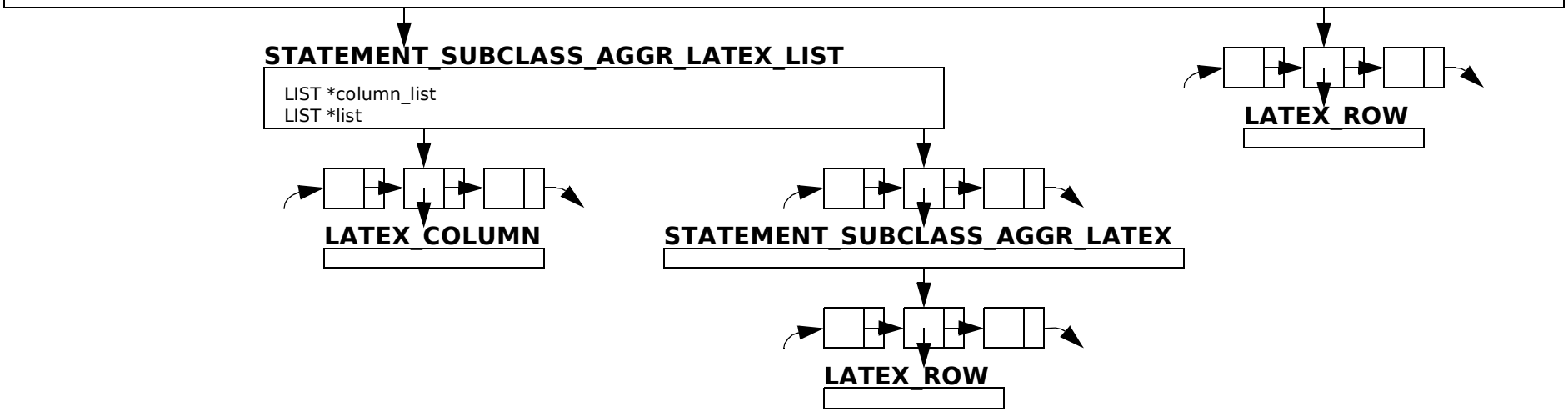
INCOME_STATEMENT_SUBCLASS_OMIT_LATEX

```
/* Usage */
INCOME_STATEMENT_SUBCLASS_OMIT_LATEX *
    income_statement_subclass_omit_latex_new(
        STATEMENT *statement,
        LIST *statement_prior_year_list(),
        double element_net_income(),
        char *income_statement_net_income_percent_of_income_display(),
        char *income_statement_net_income_label() );

/* Process */
INCOME_STATEMENT_SUBCLASS_OMIT_LATEX *
    income_statement_subclass_omit_latex_calloc(
        void );

STATEMENT_SUBCLASS_OMIT_LATEX_LIST *
    statement_subclass_omit_latex_list =
        statement_subclass_omit_latex_list_new(
            statement->element_statement_list,
            statement->statement_prior_year_list );

LIST *row_list =
    statement_subclass_omit_latex_list_row_list(
        statement_subclass_omit_latex_list );

list_set(
    row_list,
    LATEX_ROW *income_statement_latex_net_income_row(
        statement_prior_year_list,
        element_net_income,
        income_statement_net_income_percent_of_income_display,
        income_statement_net_income_label,
        statement_subclass_omit_latex_list->column_list
            /* latex_column_list */,
        1 /* empty_cell_count */ );

/* Attributes */
STATEMENT_SUBCLASS_OMIT_LATEX_LIST *
    statement_subclass_omit_latex_list;
LIST *row_list;
```



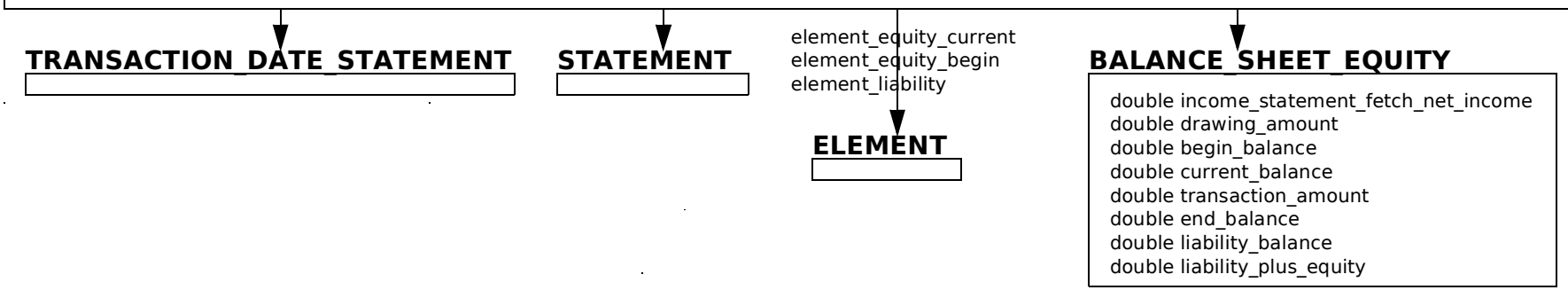
INCOME_STATEMENT_SUBCLASS_AGGR_LATEX

```
/* Usage */
INCOME_STATEMENT_SUBCLASS_AGGR_LATEX *
    income_statement_subclass_aggr_latex_new(
        STATEMENT *statement,
        LIST *statement_prior_year_list(),
        double element_net_income(),
        char *income_statement_net_income_percent_of_income_display(),
        char *income_statement_net_income_label() );

/* Process */
INCOME_STATEMENT_SUBCLASS_AGGR_LATEX *
    income_statement_subclass_aggr_latex_calloc(
        void );

STATEMENT_SUBCLASS_AGGR_LATEX_LIST *
    statement_subclass_aggr_latex_list =
        statement_subclass_aggr_latex_list_new(
            statement->element_statement_list,
            statement->statement_prior_year_list );

LIST *row_list =
    statement_subclass_aggr_latex_list_row_list(
        statement_subclass_aggr_latex_list );

list_set(
    row_list,
    LATEX_ROW *income_statement_latex_net_income_row(
        statement_prior_year_list,
        element_net_income,
        income_statement_net_income_percent_of_income_display,
        income_statement_net_income_label,
        statement_subclass_aggr_latex_list->column_list
            /* latex_column_list */,
        1 /* empty_cell_count */ );

/* Attributes */
STATEMENT_SUBCLASS_AGGR_LATEX_LIST *
    statement_subclass_aggr_latex_list;
LIST *row_list;
```



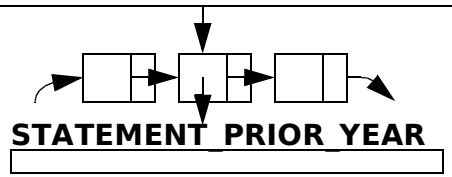
BALANCE_SHEET (1)

```
/* Usage */
BALANCE_SHEET *balance_sheet_fetch(
    char *argv_0,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *process_name,
    char *appaserver_parameter_data_directory,
    char *as_of_date_string,
    int prior_year_count,
    char *subclassification_option_string,
    char *output_medium_string );

/* Process */
BALANCE_SHEET *balance_sheet_calloc( void );

enum statement_subclassification_option =
    statement_resolve_subclassification_option(
        subclassification_option_string );

enum statement_output_medium =
    statement_resolve_output_medium(
        output_medium_string );

TRANSACTION_DATE_STATEMENT *
    transaction_date_statement =
        transaction_date_statement_new(
            as_of_date_string );

if ( !transaction_date_statement->
        transaction_date_as_of )
{
    return NULL;
}

LIST *balance_sheet_element_name_list(
    char *ELEMENT_ASSET,
    char *ELEMENT_LIABILITY );

STATEMENT *statement =
    statement_fetch(
        application_name,
        process_name,
        prior_year_count,
        balance_sheet_element_name_list(),
        transaction_date_statement->
            transaction_date_begin_date_string,
        transaction_date_statement->
            end_date_time_string,
        0 /* not fetch_transaction */ );

ELEMENT *element_equity_current =
    element_statement_fetch(
        ELEMENT_EQUITY,
        transaction_date_statement->
            end_date_time_string,
        1 /* fetch_subclassification_list */,
        1 /* fetch_account_list */,
        1 /* fetch_journal_latest */,
        0 /* not fetch_transaction */ );

double element_equity_current->sum =
    element_sum(
        element_equity_current );

ELEMENT *element_equity_begin =
    element_statement_fetch(
        ELEMENT_EQUITY,
        transaction_date_statement->
            prior_end_date_time_string,
        1 /* fetch_subclassification_list */,
        1 /* fetch_account_list */,
        1 /* fetch_journal_latest */,
        0 /* not fetch_transaction */ );

if ( element_equity_begin )
{
    double element_equity_begin->sum =
        element_sum(
            element_equity_begin );
}

ELEMENT *element_liability =
    element_seek(
        ELEMENT_LIABILITY,
        statement->element_statement_list );

double income_statement_fetch_net_income(
    transaction_date_statement->
        end_date_time_string );

double balance_sheet_drawing_amount(
    const char *JOURNAL_TABLE,
    const char *ACCOUNT_DRAWING_KEY,
    char *transaction_date_statement->
        end_date_time_string );

BALANCE_SHEET_EQUITY *balance_sheet_equity =
    balance_sheet_equity_new(
        element_equity_begin,
        element_equity_current,
        element_liability,
        income_statement_fetch_net_income(),
        balance_sheet_drawing_amount() );
```



BALANCE_SHEET (2)

```
/* Process (continued) */
if ( prior_year_count )
{
   LIST *statement_prior_year_list(
      balance_sheet_element_name_list(),
      transaction_date_statement->
         end_date_time_string,
      prior_year_count,
      statement );
}

char *income_statement_net_income_label(
    char *argv_0 );

if (statement_subclassification_option ==
   statement_subclassification_omit )
{
   element_list_account_statement_list_set(
      statement->
         element_statement_list );
}

if ( statement_output_medium ==
      statement_output_PDF )
{
   BALANCE_SHEET_LATEX *
      balance_sheet_latex =
         balance_sheet_latex_new(
            application_name,
            process_name,
            appaserver_parameter_data_directory,
            statement_subclassification_option,
            statement,
            balance_sheet_equity,
            statement_prior_year_list(),
            income_statement_net_income_label(),
            getpid() /* process_id */ );
}
else
if ( statement_output_medium ==
      statement_output_table )
{
   void element_account_transaction_count_set(
      statement->element_statement_list,
      transaction_date_statement->
         transaction_date_begin_date_string,
      transaction_date_statement->
         end_date_time_string );

   void element_account_action_string_set(
      statement->element_statement_list,
      application_name,
      session_key,
      login_name,
      role_name,
      transaction_date_statement->
         transaction_date_begin_date_string,
      transaction_date_statement->
         end_date_time_string );
   BALANCE_SHEET_HTML *
      balance_sheet_html =
         balance_sheet_html_new(
            statement_subclassification_option,
            statement,
            balance_sheet_equity,
            statement_prior_year_list,
            income_statement_net_income_label() );
}

/* Attributes */
enum statement_subclassification_option
   statement_subclassification_option;
enum statement_output_medium
   statement_output_medium;
TRANSACTION_DATE_STATEMENT *
   transaction_date_statement;
LIST *element_name_list;
STATEMENT *statement;
ELEMENT *element_equity_current;
ELEMENT *element_equity_begin;
ELEMENT *element_liability;
double income_statement_fetch_net_income;
double drawing_amount;
BALANCE_SHEET_EQUITY *balance_sheet_equity;
LIST *statement_prior_year_list;
char *income_statement_net_income_label;
BALANCE_SHEET_LATEX *balance_sheet_latex;
BALANCE_SHEET_HTML *balance_sheet_html;
```



STATEMENT_PRIOR_YEAR

BALANCE_SHEET_LATEX

BALANCE_SHEET_HTML

BALANCE_SHEET (3)

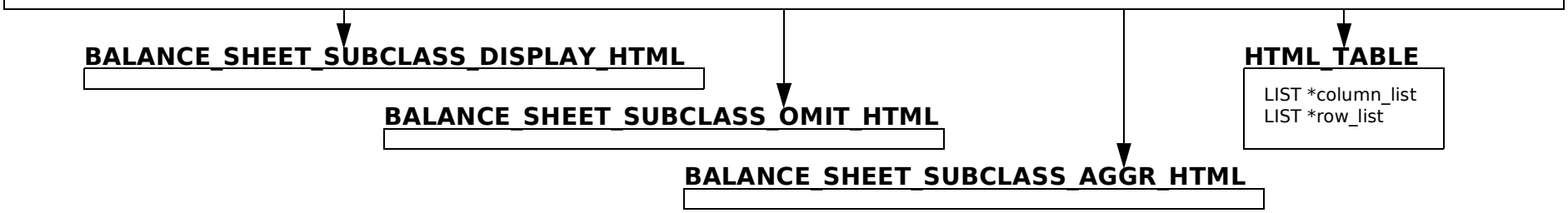
```
/* Usage */
double balance_sheet_drawing_amount(
    const char *JOURNAL_TABLE,
    const char *ACCOUNT_DRAWING_KEY,
    char *transaction_end_date_time_string );

/* Process */
char *account_drawing_string(
    account_drawing_key );

if ( !account_drawing_string() ) return 0.0;

ACCOUNT_JOURNAL *account_journal_latest(
    JOURNAL_TABLE,
    account_drawing_string() /* account_name */,
    transaction_end_date_time_string,
    0 /* not fetch_transaction */ );

if ( account_journal_latest() )
    double drawing_amount =
        account_journal_latest()->balance;

/* Driver */
printf(
    "%s\n",
    balance_sheet->
      statement->
      statement_caption->
      frame_title );

if ( balance_sheet->balance_sheet_latex )
{
    void latex_table_output(
      balance_sheet->
        balance_sheet_latex->
        statement_link->
        tex_filename,
      balance_sheet_latex->
        statement_link->
        appaserver_link_working_directory,
      balance_sheet->
        balance_sheet_latex->
        statement_link->
        pdf_anchor_html,
      balance_sheet->
        balance_sheet_latex->
        latex,
      balance_sheet->
        balance_sheet_latex->
        latex_table );
}
else
if ( balance_sheet->balance_sheet_html )
{
    void statement_html_output(
      balance_sheet->
        balance_sheet_html->
        html_table,
        (char *)0 /* secondary_title */ );
}
```

BALANCE_SHEET_EQUITY

```
/* Usage */
BALANCE_SHEET_EQUITY *
    balance_sheet_equity_new(
        ELEMENT *element_equity_begin,
        ELEMENT *element_equity_current,
        ELEMENT *element_liability,
        double income_statement_fetch_net_income(),
        double balance_sheet_drawing_amount );

/* Process */
BALANCE_SHEET_EQUITY *
    balance_sheet_equity_calloc(
        void );

if ( element_equity_begin )
{
    double balance_sheet_equity_begin_balance(
        ELEMENT *element_equity_begin );
}

double balance_sheet_equity_current_balance(
    ELEMENT *element_equity_current );

double balance_sheet_equity_transaction_amount(
    double balance_sheet_drawing_amount,
    double balance_sheet_equity_current_balance(),
    double balance_sheet_equity_begin_balance() );

double balance_sheet_equity_end_balance(
    double element_equity_current_balance(),
    double income_statement_fetch_net_income );

if ( element_liability )
{
    double balance_sheet_equity_liability_balance(
        ELEMENT *element_liability );
}

double balance_sheet_equity_liability_plus_equity(
    double balance_sheet_equity_liability_balance(),
    double balance_sheet_equity_end_balance() );

/* Usage */
HTML_ROW *balance_sheet_equity_html_row(
    const char *label,
    double amount );

/* Process */

/* Usage */
LATEX_ROW *balance_sheet_equity_latex_row(
    int empty_cell_count,
    const char *label,
    double amount,
    LIST *latex_column_list );

/* Process */

/* Attributes */
double income_statement_fetch_net_income;
double drawing_amount;
double begin_balance;
double current_balance;
double transaction_amount;
double end_balance;
double liability_balance;
double liability_plus_equity;
```

BALANCE_SHEET_HTML (1)

```
/* Usage */
BALANCE_SHEET_HTML *
   balance_sheet_html_new(
      enum statement_subclassification_option
         statement_subclassification_option,
      STATEMENT *statement,
      BALANCE_SHEET_EQUITY *balance_sheet_equity,
      LIST *statement_prior_year_list,
      char *income_statement_net_income_label() );

/* Process */
BALANCE_SHEET_HTML *balance_sheet_html_calloc(
   void );

if (   statement_subclassification_option ==
       statement_subclassification_display )
{
   BALANCE_SHEET_SUBCLASS_DISPLAY_HTML *
      balance_sheet_subclass_display_html =
         balance_sheet_subclass_display_html_new(
            statement->element_statement_list,
            balance_sheet_equity,
            statement_prior_year_list,
            income_statement_net_income_label );

   HTML_TABLE *balance_sheet_html_table(
      statement->
         statement_caption->
         sub_title,
         balance_sheet_subclass_display_html->
            statement_subclass_display_html->
            column_list,
         balance_sheet_subclass_display_html->row_list );
}
else
if (   statement_subclassification_option ==
       statement_subclassification_omit )
{
   BALANCE_SHEET_SUBCLASS_OMIT_HTML *
      balance_sheet_subclass_omit_html =
         balance_sheet_subclass_omit_html_new(
            statement->element_statement_list,
            balance_sheet_equity,
            statement_prior_year_list,
            income_statement_net_income_label );

   HTML_TABLE *balance_sheet_html_table(
      statement->
         statement_caption->
         sub_title,
      balance_sheet_subclass_omit_html->
         statement_subclass_omit_html->
         column_list,
      balance_sheet_subclass_omit_html->row_list );
}
else
if (   statement_subclassification_option ==
       statement_subclassification_aggregate )
{
   BALANCE_SHEET_SUBCLASS_AGGR_HTML *
      balance_sheet_subclass_aggr_html =
         balance_sheet_subclass_aggr_html_new(
            statement->element_statement_list,
            balance_sheet_equity,
            statement_prior_year_list,
            income_statement_net_income_label );

   HTML_TABLE *balance_sheet_html_table(
      statement->
         statement_caption->
         sub_title,
      balance_sheet_subclass_aggr_html->
         statement_subclass_aggr_html->
         column_list,
      balance_sheet_subclass_aggr_html->row_list );
}

/* Attributes */
BALANCE_SHEET_SUBCLASS_DISPLAY_HTML *
   balance_sheet_subclass_display_html;
BALANCE_SHEET_SUBCLASS_OMIT_HTML *
   balance_sheet_subclass_omit_html;
BALANCE_SHEET_SUBCLASS_AGGR_HTML *
   balance_sheet_subclass_aggr_html;
HTML_TABLE *html_table;
```

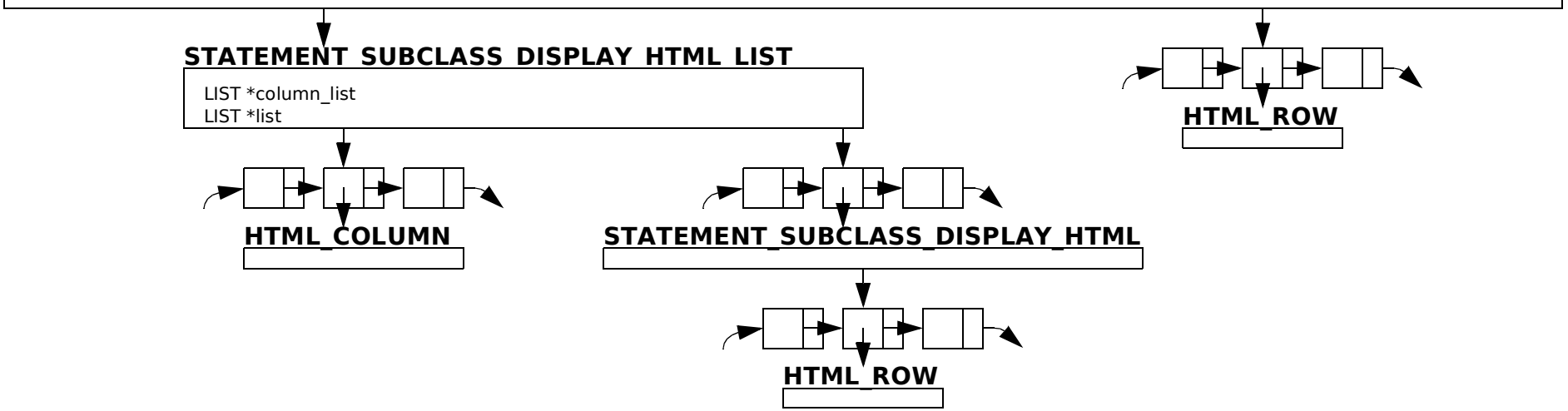


BALANCE_SHEET_HTML (2)

```
/* Usage */
HTML_ROW *balance_sheet_html_net_income_row(
    double income_statement_fetch_net_income(),
    char *income_statement_net_income_label() );

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

char *statement_cell_label_datum(
    income_statement_net_income_label ),

HTML_CELL *html_cell_new(
    statement_cell_label_datum(),
    0 /* not large_boolean */,
    0 /* not bold_boolean */ );

list_set( cell_list, html_cell_new() );

char *string_commas_money(
    income_statement_fetch_net_income );

list_set(
    cell_list,
    html_cell_new(
        strdup( string_commas_money() ),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

HTML_ROW *html_row_new( cell_list );

/* Usage */
HTML_TABLE *balance_sheet_html_table(
    char *statement_caption_sub_title,
    LIST *column_list,
    LIST *row_list );

/* Process */
HTML_TABLE *html_table =
    html_table_new(
        (char *)0 /* title */,
        statement_caption_sub_title,
        (char *)0 /* sub_sub_title */ );

html_table->column_list = column_list;
html_table->row_list = row_list;
```

BALANCE_SHEET_SUBCLASS_DISPLAY_HTML (1)

```
/* Usage */
BALANCE_SHEET_SUBCLASS_DISPLAY_HTML *
    balance_sheet_subclass_display_html_new(
       LIST *element_statement_list,
       BALANCE_SHEET_EQUITY *balance_sheet_equity,
       LIST *statement_prior_year_list(),
       char *income_statement_net_income_label() );

/* Process */
BALANCE_SHEET_SUBCLASS_DISPLAY_HTML *
    balance_sheet_subclass_display_html_calloc(
       void );

STATEMENT_SUBCLASS_DISPLAY_HTML_LIST *
    statement_subclass_display_html_list =
        statement_subclass_display_html_list_new(
           element_statement_list,
           statement_prior_year_list );

LIST *row_list =
    LIST *statement_subclass_display_html_list_row_list(
       statement_subclass_display_html_list );

list_set_list(
    row_list,
    LIST *balance_sheet_subclass_display_html_equity_row_list(
       balance_sheet_equity,
       income_statement_net_income_label ) );
```

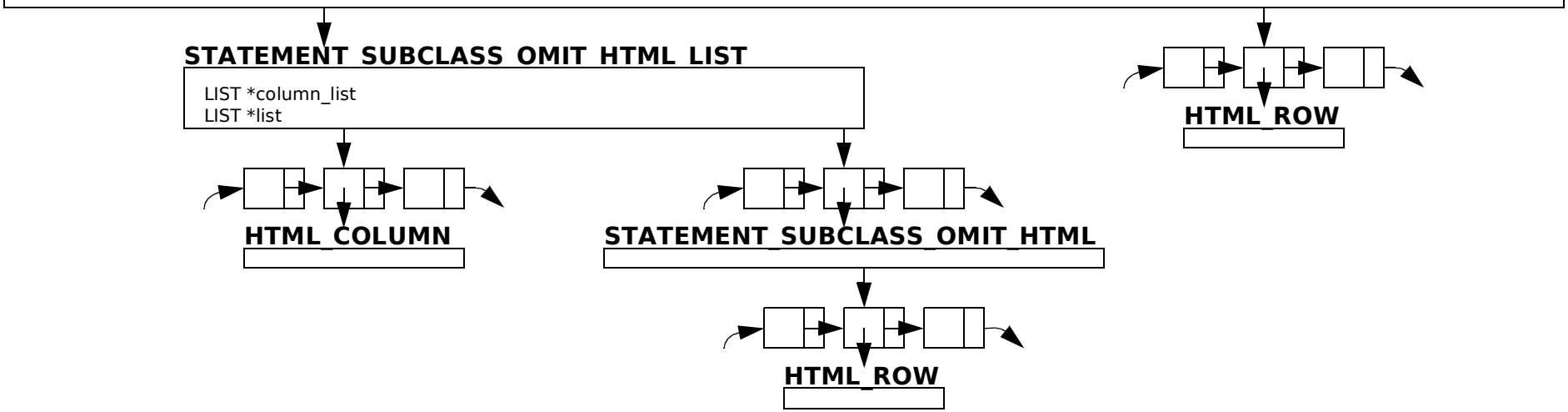


BALANCE_SHEET_SUBCLASS_DISPLAY_HTML (2)

```
/* Usage */
LIST *balance_sheet_subclass_display_html_equity_row_list(
    BALANCE_SHEET_EQUITY *balance_sheet_equity,
    char *income_statement_net_income_label() );

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

list_set(
    row_list,
    HTML_ROW *balance_sheet_subclass_display_html_equity_begin_row(
       char *BALANCE_SHEET_BEGIN_BALANCE_LABEL,
       double balance_sheet_equity->begin_balance ) );

if ( balance_sheet_equity->transaction_amount )
{
    list_set(
       row_list,
       HTML_ROW *balance_sheet_equity_html_row(
          BALANCE_SHEET_TRANSACTION_AMOUNT_LABEL,
          balance_sheet_equity->transaction_amount ) );
}

if ( balance_sheet_equity->drawing_amount )
{
    list_set(
       row_list,
       HTML_ROW *balance_sheet_equity_html_row(
          BALANCE_SHEET_DRAWING_LABEL,
          -balance_sheet_equity->amount ) );
}

list_set(
    row_list,
    HTML_ROW *balance_sheet_html_net_income_row(
       balance_sheet_equity->income_statement_fetch_net_income,
       income_statement_net_income_label ) );

list_set(
    row_list,
    HTML_ROW *balance_sheet_subclass_display_html_equity_end_row(
       char *BALANCE_SHEET_END_BALANCE_LABEL,
       double balance_sheet_equity->end_balance ) );

list_set(
    row_list,
    HTML_ROW *balance_sheet_subclass_display_html_liability_plus_equity_row(
       char *BALANCE_SHEET_LIABILITY_PLUS_EQUITY_LABEL,
       double balance_sheet_equity->liability_plus_equity ) );
```

BALANCE_SHEET_SUBCLASS_OMIT_HTML (1)

```
/* Usage */
BALANCE_SHEET_SUBCLASS_OMIT_HTML *
    balance_sheet_subclass_omit_html_new(
       LIST *element_statement_list,
       BALANCE_SHEET_EQUITY *balance_sheet_equity,
       LIST *statement_prior_year_list(),
       char *income_statement_net_income_label() );

/* Process */
BALANCE_SHEET_SUBCLASS_OMIT_HTML *
    balance_sheet_subclass_display_html_calloc(
       void );

STATEMENT_SUBCLASS_OMIT_HTML_LIST *
    statement_subclass_omit_html_list =
       statement_subclass_omit_html_list_new(
          element_statement_list,
          statement_prior_year_list );

LIST *row_list =
    LIST *statement_subclass_omit_html_list_row_list(
       statement_subclass_omit_html_list );

list_set_list(
    row_list,
    LIST *balance_sheet_subclass_omit_html_equity_row_list(
       balance_sheet_equity,
       income_statement_net_income_label ) );
```

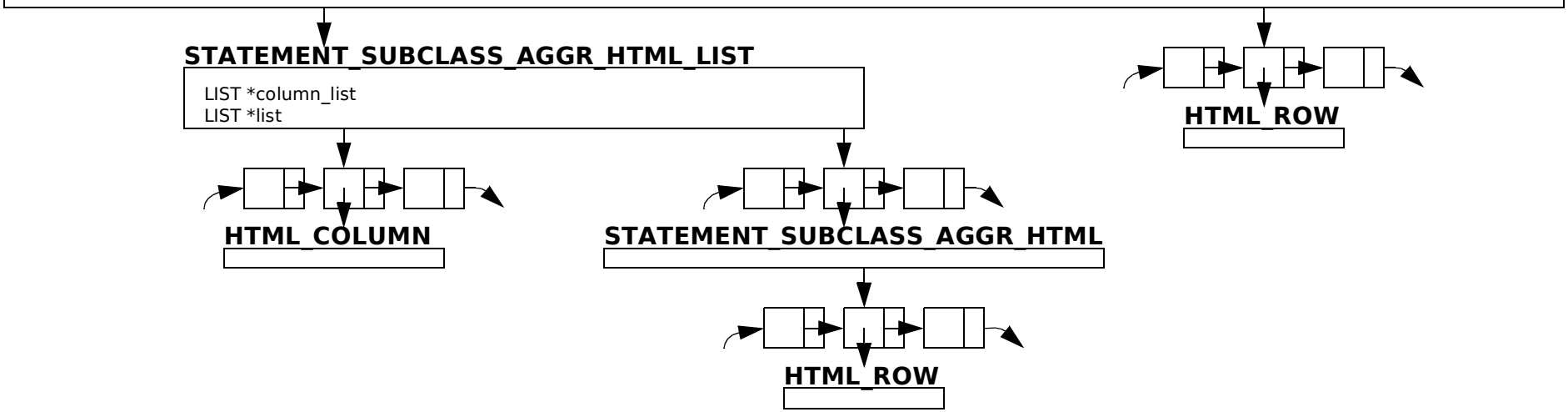


BALANCE_SHEET_SUBCLASS_OMIT_HTML (2)

```
/* Usage */
LIST *balance_sheet_subclass_omit_html_equity_row_list(
    BALANCE_SHEET_EQUITY *balance_sheet_equity,
    char *income_statement_net_income_label() );

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

list_set(
    row_list,
    HTML_ROW *balance_sheet_subclass_omit_html_equity_begin_row(
       char *BALANCE_SHEET_BEGIN_BALANCE_LABEL,
       double balance_sheet_equity->begin_balance ) );

if ( balance_sheet_equity->transaction_amount )
{
    list_set(
       row_list,
       HTML_ROW *balance_sheet_equity_html_row(
          BALANCE_SHEET_TRANSACTION_AMOUNT_LABEL,
          balance_sheet_equity->transaction_amount ) );
}

if ( balance_sheet_equity->drawing_amount )
{
    list_set(
       row_list,
       HTML_ROW *balance_sheet_equity_html_row(
          BALANCE_SHEET_DRAWING_LABEL,
          -balance_sheet_equity->drawing_amount ) );
}

list_set(
    row_list,
    HTML_ROW *balance_sheet_html_net_income_row(
       balance_sheet_equity->income_statement_fetch_net_income,
       income_statement_net_income_label ) );

list_set(
    row_list,
    HTML_ROW *balance_sheet_subclass_omit_html_equity_end_row(
       char *BALANCE_SHEET_END_BALANCE_LABEL,
       double balance_sheet_equity->end_balance ) );

list_set(
    row_list,
    HTML_ROW *balance_sheet_subclass_omit_html_liability_plus_equity_row(
       char *BALANCE_SHEET_LIABILITY_PLUS_EQUITY_LABEL,
       double balance_sheet_equity->liability_plus_equity ) );
```

BALANCE_SHEET_SUBCLASS_AGGR_HTML (1)

```
/* Usage */
BALANCE_SHEET_SUBCLASS_AGGR_HTML *
    balance_sheet_subclass_aggr_html_new(
       LIST *element_statement_list,
       BALANCE_SHEET_EQUITY *balance_sheet_equity,
       LIST *statement_prior_year_list(),
       char *income_statement_net_income_label() );

/* Process */
BALANCE_SHEET_SUBCLASS_OMIT_HTML *
    balance_sheet_subclass_display_html_calloc(
       void );

STATEMENT_SUBCLASS_OMIT_HTML_LIST *
    statement_subclass_aggr_html_list =
       statement_subclass_aggr_html_list_new(
          element_statement_list,
          statement_prior_year_list );

LIST *row_list =
    LIST *statement_subclass_aggr_html_list_row_list(
       statement_subclass_aggr_html_list );

list_set_list(
    row_list,
    LIST *balance_sheet_subclass_aggr_html_equity_row_list(
       balance_sheet_equity,
       income_statement_net_income_label ) );

/* Attributes */
STATEMENT_SUBCLASS_AGGR_HTML_LIST *
    statement_subclass_aggr_html_list;
LIST *row_list;
```



BALANCE_SHEET_SUBCLASS_AGGR_HTML (2)

```
/* Usage */
LIST *balance_sheet_subclass_aggr_html_equity_row_list(
    BALANCE_SHEET_EQUITY *balance_sheet_equity,
    char *income_statement_net_income_label() );

/* Process */
list_set(
    row_list,
    HTML_ROW *balance_sheet_subclass_aggr_html_equity_begin_row(
       char *BALANCE_SHEET_BEGIN_BALANCE_LABEL,
       double balance_sheet_equity->begin_balance ) );

if ( balance_sheet_equity->transaction_amount )
{
    list_set(
       row_list,
       HTML_ROW *balance_sheet_equity_amount_row(
          BALANCE_SHEET_TRANSACTION_AMOUNT_LABEL,
          balance_sheet_equity->transaction_amount ) );
}

if ( balance_sheet_equity->drawing_amount )
{
    list_set(
       row_list,
       HTML_ROW *balance_sheet_equity_amount_row(
          BALANCE_SHEET_DRAWING_LABEL,
          -balance_sheet_equity->drawing_amount ) );
}

list_set(
    row_list,
    HTML_ROW *balance_sheet_html_net_income_row(
       balance_sheet_equity->income_statement_fetch_net_income,
       income_statement_net_income_label ) );

list_set(
    row_list,
    HTML_ROW *balance_sheet_subclass_aggr_html_equity_end_row(
       char *BALANCE_SHEET_END_BALANCE_LABEL,
       double balance_sheet_equity->end_balance ) );

list_set(
    row_list,
    HTML_ROW *balance_sheet_subclass_aggr_html_liability_plus_equity_row(
       char *BALANCE_SHEET_LIABILITY_PLUS_EQUITY_LABEL,
       double balance_sheet_equity->liability_plus_equity ) );
```

BALANCE_SHEET_LATEX (1)

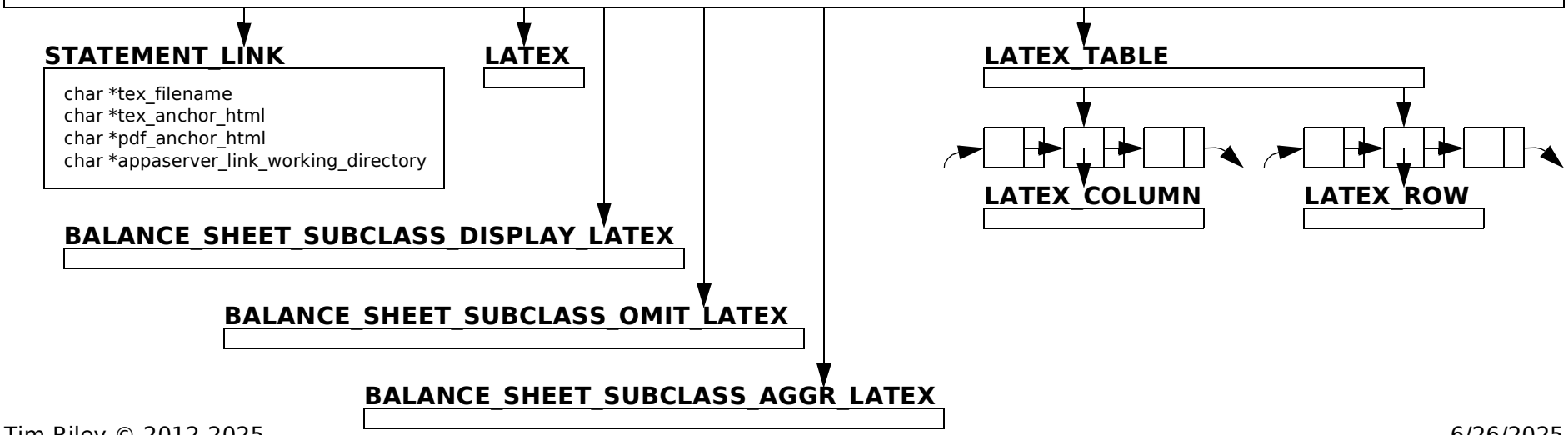
```
/* Usage */
BALANCE_SHEET_LATEX *
    balance_sheet_latex_new(
        char *application_name,
        char *process_name,
        char *appaserver_parameter_data_directory,
        enum statement_subclassification_option
            statement_subclassification_option,
        STATEMENT *statement,
        BALANCE_SHEET_EQUITY *balance_sheet_equity,
        LIST *statement_prior_year_list,
        char *income_statement_net_income_label(),
        pid_t process_id );

/* Process */
BALANCE_SHEET_LATEX *balance_sheet_latex_calloc(
    void );

STATEMENT_LINK *statement_link =
    statement_link_new(
        application_name,
        process_name,
        appaserver_parameter_data_directory,
        statement->transaction_date_begin_date_string,
        statement->end_date_time_string,
        process_id );

LATEX *latex =
    latex_new(
        statement_link->tex_filename,
        statement_link->
            appaserver_link_working_directory,
        statement->pdf_landscape_boolean,
        statement->
            statement_caption->
            logo_filename );

if (    statement_subclassification_option ==
        statement_subclassification_display )
{
    BALANCE_SHEET_SUBCLASS_DISPLAY_LATEX *
        balance_sheet_subclass_display_latex =
            balance_sheet_subclass_display_latex_new(
                statement->element_statement_list,
                balance_sheet_equity,
                statement_prior_year_list,
                income_statement_net_income_label );

    LATEX_TABLE *latex_table =
        LATEX_TABLE *balance_sheet_latex_table(
            statement->statement_caption->combined,
            balance_sheet_subclass_display_latex->
                statement_subclass_display_latex_list->
                column_list,
            balance_sheet_subclass_display_latex->
                row_list );
}
else
if (    statement_subclassification_option ==
        statement_subclassification_omit )
{
    BALANCE_SHEET_SUBCLASS_OMIT_LATEX *
        balance_sheet_subclass_omit_latex =
            balance_sheet_subclass_omit_latex_new(
                statement->element_statement_list,
                balance_sheet_equity,
                statement_prior_year_list,
                income_statement_net_income_label );

    LATEX_TABLE *latex_table =
        LATEX_TABLE *balance_sheet_latex_table(
            statement->statement_caption->combined,
            balance_sheet_subclass_omit_latex->
                statement_subclass_omit_latex_list->
                column_list,
            balance_sheet_subclass_omit_latex->
                row_list ) );
}
else
if (    statement_subclassification_option ==
        statement_subclassification_aggregate )
{
    BALANCE_SHEET_SUBCLASS_AGGR_LATEX *
        balance_sheet_subclass_aggr_latex =
            balance_sheet_subclass_aggr_latex_new(
                statement->element_statement_list,
                balance_sheet_equity,
                statement_prior_year_list,
                income_statement_net_income_label );

    LATEX_TABLE *latex_table =
        LATEX_TABLE *balance_sheet_latex_table(
            statement->statement_caption->combined,
            balance_sheet_subclass_aggr_latex->
                statement_subclass_aggr_latex_list->
                column_list,
            balance_sheet_subclass_aggr_latex->
                row_list ) );
}
```

BALANCE_SHEET_LATEX (2)

```
/* Usage */
LATEX_TABLE *balance_sheet_latex_table(
    char *statement_caption_combined,
    LIST *column_list,
    LIST *row_list );

/* Process */
LATEX_TABLE *latex_table =
    latex_table_new(
       statement_caption_combined /* title */,
       column_list,
       row_list );

/* Usage */
LATEX_ROW *balance_sheet_latex_net_income_row(
    double income_statement_fetch_net_income(),
    char *income_statement_net_income_label(),
    int empty_cell_count,
    LIST *latex_column_list );

/* Process */
list_rewind( latex_column_list );

LIST *cell_list = list_new();

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
list_next( latex_column_list );
```

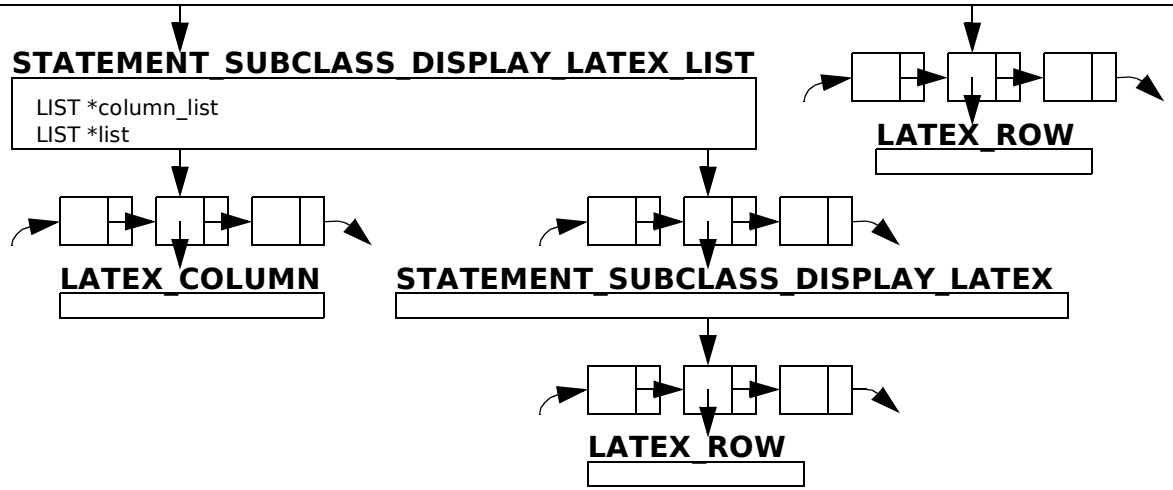
```
char *statement_cell_label_datum(
    income_statement_net_income_label ),

LATEX_CELL *latex_cell_small_new(
    latex_column,
    0 /* not first_row_boolean */,
    statement_cell_label_datum() );

list_set( cell_list, latex_cell_small_new() );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
list_next( latex_column_list );

char *string_commas_money(
    income_statement_fetch_net_income );

list_set(
    cell_list,
    latex_cell_small_new(
       latex_column,
       0 /* not first_row_boolean */,
       strdup( string_commas_money() ) ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
list_next( latex_column_list );

for ( i = 0;
```

```
      i < empty_cell_count;
      i++ )
{
    list_set(
       cell_list,
       latex_cell_small_new(
          latex_column,
          0 /* not first_row_boolean */,
          (char *)0 /* datum */ ) );

    LATEX_COLUMN *latex_column =
       list_get( latex_column_list );
    list_next( latex_column_list );
}

LATEX_ROW *latex_row_new( cell_list );

/* Attributes */
STATEMENT_LINK *statement_link;
LATEX *latex;
BALANCE_SHEET_SUBCLASS_DISPLAY_LATEX *
    balance_sheet_subclass_display_latex;
BALANCE_SHEET_SUBCLASS_OMIT_LATEX *
    balance_sheet_subclass_omit_latex;
BALANCE_SHEET_SUBCLASS_AGGR_LATEX *
    balance_sheet_subclass_aggr_latex;
LATEX_TABLE *latex_table;
```



BALANCE_SHEET_SUBCLASS_DISPLAY_LATEX (1)

```
/* Usage */
BALANCE_SHEET_SUBCLASS_DISPLAY_LATEX *
    balance_sheet_subclass_display_latex_new(
       LIST *element_statement_list,
       BALANCE_SHEET_EQUITY *balance_sheet_equity,
       LIST *statement_prior_year_list(),
       char *income_statement_net_income_label() );

/* Process */
BALANCE_SHEET_SUBCLASS_DISPLAY_LATEX *
    balance_sheet_subclass_display_latex_calloc(
       void );

STATEMENT_SUBCLASS_DISPLAY_LATEX_LIST *
    statement_subclass_display_latex_list =
       statement_subclass_display_latex_list_new(
          element_statement_list,
          statement_prior_year_list );

LIST *row_list =
    LIST *statement_subclass_display_latex_list_row_list(
          statement_subclass_display_latex_list );

list_set_list(
    row_list,
    LIST *balance_sheet_subclass_display_latex_equity_row_list(
       list_length( statement_prior_year_list )
          /* statement_prior_year_list_length */,
       balance_sheet_equity->begin_balance,
       balance_sheet_equity->transaction_amount,
       balance_sheet_equity->drawing_amount,
       balance_sheet_equity->end_balance,
       balance_sheet_equity->liability_plus_equity,
       balance_sheet_equity->income_statement_fetch_net_income,
       income_statement_net_income_label,
       statement_subclass_display_latex_list->column_list
          /* latex_column_list */ ) );

/* Attributes */
STATEMENT_SUBCLASS_DISPLAY_LATEX_LIST *
    statement_subclass_display_latex_list;
LIST *row_list;
```

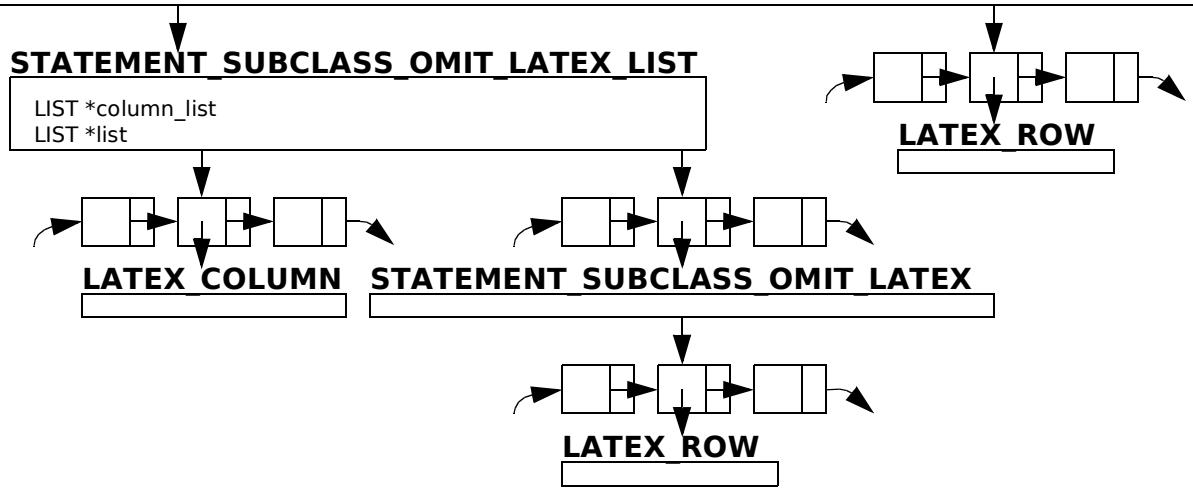


BALANCE_SHEET_SUBCLASS_DISPLAY_LATEX (2)

```
/* Usage */
LIST *balance_sheet_subclass_display_latex_equity_row_list(
    int statement_prior_year_list_length,
    double balance_sheet_equity_begin_balance,
    double balance_sheet_equity_transaction_amount,
    double balance_sheet_equity_drawing_amount,
    double balance_sheet_equity_end_balance,
    double balance_sheet_equity_liability_plus_equity,
    double income_statement_fetch_net_income(),
    char *income_statement_net_income_label(),
    LIST *statement_subclass_display_latex_list->column_list
      /* latex_column_list */ );

/* Process */
LIST *row_list;

list_set(
    row_list,
    LATEX_ROW *balance_sheet_subclass_display_latex_equity_begin_row(
      int statement_prior_year_list_length,
      char *BALANCE_SHEET_EQUITY_BEGIN_BALANCE_LABEL,
      double balance_sheet_equity_begin_balance,
      LIST *latex_column_list ) );

if ( balance_sheet_equity_transaction_amount )
{
    list_set(
      row_list,
      LATEX_ROW *balance_sheet_equity_latex_row(
        statement_prior_year_list_length /* empty_cell_count */,
        BALANCE_SHEET_TRANSACTION_AMOUNT_LABEL,
        balance_sheet_equity_transaction_amount,
        latex_column_list ) );
}

if ( balance_sheet_equity_drawing_amount )
{
    list_set(
      row_list,
      LATEX_ROW *balance_sheet_equity_latex_row(
        statement_prior_year_list_length /* empty_cell_count */,
        BALANCE_SHEET_DRAWING_LABEL,
        -balance_sheet_equity_drawing_amount,
        latex_column_list ) );
}

list_set(
    row_list,
    LATEX_ROW *balance_sheet_latex_net_income_row(
      income_statement_fetch_net_income,
      income_statement_net_income_label,
      statement_prior_year_list_length + 3
        /* empty_cell_count */,
      latex_column_list ) );

list_set(
    row_list,
    LATEX_ROW *balance_sheet_subclass_display_latex_equity_end_row(
      int statement_prior_year_list_length,
      char *BALANCE_SHEET_END_BALANCE_LABEL,
      double balance_sheet_equity_end_balance,
      LIST *latex_column_list ) );

list_set(
    row_list,
    LATEX_ROW *balance_sheet_subclass_display_latex_liability_plus_equity_row(
      int statement_prior_year_list_length,
      char *BALANCE_SHEET_LIABILITY_PLUS_EQUITY_LABEL,
      double balance_sheet_equity_liability_plus_equity,
      LIST *latex_column_list ) );
```

BALANCE_SHEET_SUBCLASS_OMIT_LATEX (1)

```
/* Usage */
BALANCE_SHEET_SUBCLASS_OMIT_LATEX *
    balance_sheet_subclass_omit_latex_new(
       LIST *element_statement_list,
       BALANCE_SHEET_EQUITY *balance_sheet_equity,
       LIST *statement_prior_year_list(),
       char *income_statement_net_income_label() );

/* Process */
BALANCE_SHEET_SUBCLASS_OMIT_LATEX *
    balance_sheet_subclass_omit_latex_calloc(
       void );

STATEMENT_SUBCLASS_OMIT_LATEX_LIST *
    statement_subclass_omit_latex_list =
       statement_subclass_omit_latex_list_new(
          element_statement_list,
          statement_prior_year_list );

LIST *row_list =
    LIST *statement_subclass_omit_latex_list_row_list(
          statement_subclass_omit_latex_list );

list_set_list(
    row_list,
       LIST *balance_sheet_subclass_omit_latex_equity_row_list(
       list_length( statement_prior_year_list )
          /* statement_prior_year_list_length */,
       balance_sheet_equity->begin_balance,
       balance_sheet_equity->transaction_amount,
       balance_sheet_equity->drawing_amount,
       balance_sheet_equity->end_balance,
       balance_sheet_equity->liability_plus_equity,
       balance_sheet_equity->income_statement_fetch_net_income,
       income_statement_net_income_label,
       statement_subclass_omit_latex_list->column_list
          /* latex_column_list */ ) );

/* Attributes */
STATEMENT_SUBCLASS_OMIT_LATEX_LIST *
    statement_subclass_omit_latex_list;
LIST *row_list;
```

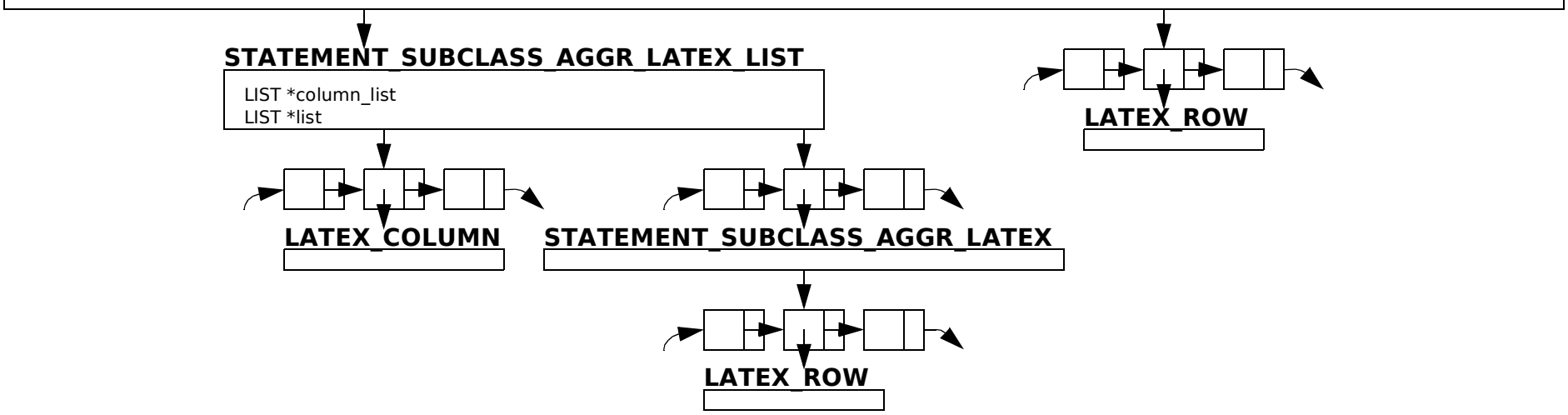


BALANCE_SHEET_SUBCLASS_OMIT_LATEX (2)

```
/* Usage */
LIST *balance_sheet_subclass_omit_latex_equity_row_list(
    int statement_prior_year_list_length,
    double balance_sheet_equity_begin_balance,
    double balance_sheet_equity_transaction_amount,
    double balance_sheet_equity_drawing_amount,
    double balance_sheet_equity_end_balance,
    double balance_sheet_equity_liability_plus_equity,
    double income_statement_fetch_net_income(),
    char *income_statement_net_income_label(),
    LIST *statement_subclass_omit_latex_list->column_list
      /* latex_column_list */ );

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

list_set(
    row_list,
    LATEX_ROW *balance_sheet_subclass_omit_latex_equity_begin_row(
      int statement_prior_year_list_length,
      char *BALANCE_SHEET_BEGIN_BALANCE_LABEL,
      double balance_sheet_equity_begin_balance,
      LIST *latex_column_list ) );

if ( balance_sheet_equity_transaction_amount )
{
    list_set(
      row_list,
      LATEX_ROW *balance_sheet_equity_latex_row(
        statement_prior_year_list_length /* empty_cell_count */,
        BALANCE_SHEET_TRANSACTION_AMOUNT_LABEL,
        balance_sheet_equity_transaction_amount,
        latex_column_list ) );
}

if ( balance_sheet_equity_drawing_amount )
{
    list_set(
      row_list,
      LATEX_ROW *balance_sheet_equity_latex_row(
        statement_prior_year_list_length /* empty_cell_count */,
        BALANCE_SHEET_DRAWING_LABEL,
        -balance_sheet_equity_drawing_amount,
        latex_column_list ) );
}

list_set(
    row_list,
    LATEX_ROW *balance_sheet_latex_net_income_row(
      income_statement_fetch_net_income,
      income_statement_net_income_label,
      statement_prior_year_list_length + 2
        /* empty_cell_count */,
      latex_column_list ) );

list_set(
    row_list,
    LATEX_ROW *balance_sheet_subclass_omit_latex_equity_end_row(
      int statement_prior_year_list_length,
      char *BALANCE_SHEET_END_BALANCE_LABEL,
      double balance_sheet_equity_end_balance,
      LIST *latex_column_list ) );

list_set(
    row_list,
    LATEX_ROW *balance_sheet_subclass_omit_latex_liability_plus_equity_row(
      int statement_prior_year_list_length,
      char *BALANCE_SHEET_LIABILITY_PLUS_EQUITY_LABEL,
      double balance_sheet_equity_liability_plus_equity,
      LIST *latex_column_list ) );
```

BALANCE_SHEET_SUBCLASS_AGGR_LATEX (1)

```
/* Usage */
BALANCE_SHEET_SUBCLASS_AGGR_LATEX *
    balance_sheet_subclass_aggr_latex_new(
        LIST *element_statement_list,
        BALANCE_SHEET_EQUITY *balance_sheet_equity,
        LIST *statement_prior_year_list(),
        char *income_statement_net_income_label() );

/* Process */
BALANCE_SHEET_SUBCLASS_AGGR_LATEX *
    balance_sheet_subclass_aggr_latex_calloc(
        void );

STATEMENT_SUBCLASS_AGGR_LATEX_LIST *
    statement_subclass_aggr_latex_list =
        statement_subclass_aggr_latex_list_new(
            element_statement_list,
            statement_prior_year_list );

LIST *row_list =
    LIST *statement_subclass_aggr_latex_list_row_list(
        statement_subclass_aggr_latex_list );

list_set_list(
    row_list,
    LIST *balance_sheet_subclass_aggr_latex_equity_row_list(
        list_length( statement_prior_year_list )
            /* statement_prior_year_list_length */,
        balance_sheet_equity->begin_balance,
        balance_sheet_equity->transaction_amount,
        balance_sheet_equity->drawing_amount,
        balance_sheet_equity->end_balance,
        balance_sheet_equity->liability_plus_equity,
        balance_sheet_equity->income_statement_fetch_net_income,
        income_statement_net_income_label,
        statement_subclass_aggr_latex->column_list
            /* latex_column_list */ ) );

/* Attributes */
STATEMENT_SUBCLASS_AGGR_LATEX_LIST *
    statement_subclass_aggr_latex_list;
LIST *row_list;
```

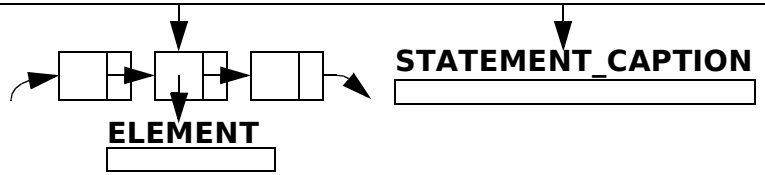


BALANCE_SHEET_SUBCLASS_AGGR_LATEX (2)

```
/* Usage */
LIST *balance_sheet_subclass_aggr_latex_equity_row_list(
    int statement_prior_year_list_length,
    double balance_sheet_equity_begin_balance,
    double balance_sheet_equity_transaction_amount,
    double balance_sheet_equity_drawing_amount,
    double balance_sheet_equity_end_balance,
    double balance_sheet_equity_liability_plus_equity,
    double income_statement_fetch_net_income(),
    char *income_statement_net_income_label(),
    LIST *statement_subclass_aggr_latex_list->column_list
      /* latex_column_list */  );

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

list_set(
    row_list,
    LATEX_ROW *balance_sheet_subclass_aggr_latex_equity_begin_row(
      int statement_prior_year_list_length,
      char *BALANCE_SHEET_BEGIN_BALANCE_LABEL,
      double balance_sheet_equity_begin_balance,
      LIST *latex_column_list ) );

if ( balance_sheet_equity_transaction_amount )
{
    list_set(
      row_list,
      LATEX_ROW *balance_sheet_equity_latex_row(
        statement_prior_year_list_length + 2 /* empty_cell_count */,
        BALANCE_SHEET_TRANSACTION_AMOUNT_LABEL,
        balance_sheet_equity_transaction_amount,
        latex_column_list ) );
}

if ( balance_sheet_equity_drawing_amount )
{
    list_set(
      row_list,
      LATEX_ROW *balance_sheet_equity_latex_row(
        statement_prior_year_list_length + 2 /* empty_cell_count */,
        BALANCE_SHEET_DRAWING_LABEL,
        -balance_sheet_equity_drawing_amount,
        latex_column_list ) );
}

list_set(
    row_list,
    LATEX_ROW *balance_sheet_latex_net_income_row(
      income_statement_fetch_net_income,
      income_statement_net_income_label,
      statement_prior_year_list_length + 2
        /* empty_cell_count */,
      latex_column_list ) );

list_set(
    row_list,
    LATEX_ROW *balance_sheet_subclass_aggr_latex_equity_end_row(
      int statement_prior_year_list_length,
      char *BALANCE_SHEET_END_BALANCE_LABEL,
      double balance_sheet_equity_end_balance,
      LIST *latex_column_list ) );

list_set(
    row_list,
    LATEX_ROW *balance_sheet_subclass_aggr_latex_liability_plus_equity_row(
      int statement_prior_year_list_length,
      char *BALANCE_SHEET_LIABILITY_PLUS_EQUITY_LABEL,
      double balance_sheet_equity_liability_plus_equity,
      LIST *latex_column_list ) );
```

STATEMENT (1)

```
/* Usage */
STATEMENT *statement_fetch(
    char *application_name /* optional */,
    char *process_name /* optional */,
    int prior_year_count,
    LIST *element_name_list,
    char *transaction_date_begin_date_string
       /* optional */,
    char *end_date_time_string,
    boolean fetch_transaction );

/* Process */
STATEMENT *statement_calloc(
    void );

LIST *element_statement_list(
    element_name_list,
    end_date_time_string,
    1 /* fetch_subclassification_list */,
    1 /* fetch_account_list */,
    1 /* fetch_journal_latest */,
    fetch_transaction );

void element_list_sum_set(
    element_statement_list() );

void element_percent_of_asset_set(
    LIST *element_statement_list );

void element_percent_of_income_set(
    LIST *element_statement_list );

boolean statement_pdf_landscape_boolean(
    int prior_year_count );

if ( application_name )
{
    STATEMENT_CAPTION *statement_caption =
        statement_caption_new(
            application_name,
            process_name,
            transaction_date_begin_date_string,
            end_date_time_string );
}

char *statement_greater_year_message(
    transaction_date_begin_date_string,
    end_date_time_string );

/* Attributes */
char *process_name;
char *transaction_date_begin_date_string;
char *end_date_time_string;
LIST *element_statement_list;
boolean pdf_landscape_boolean;
STATEMENT_CAPTION *statement_caption;
char *greater_year_message;
```



STATEMENT (2)

```
/* Usage */
char *statement_date_american(
    char *date_time_string );

/* Process */
char *column( date_string[], 0, date_time_string );

void date_convert_source_international(
    date_american[],
    date_convert_american /* destination_enum */,
    date_string ) );

/* Usage */
char *statement_cell_label_datum(
    char *name );

/* Process */
if ( !name ) return "";

char *string_initial_capital(
    buffer[],
    name );

char *strdup( buffer );

/* Usage */
boolean statement_subclassification_name_duplicated(
    char *element->element_name,
    char *subclassification->subclassification_name );

/* Process */
( strcmp(
    element_name,
    subclassification_name ) == 0 );

/* Usage */
char *statement_element_sum_name(
    char *element_name );

/* Process */
sprintf(
    static name[],
    "%s element",
    element_name );

/* Usage */
char *statement_greater_year_message(
    char *transaction_date_begin_date_string,
    char *end_date_time_string );

/* Process */
boolean date_greater_year_boolean(
    transaction_date_begin_date_string,
    end_date_time_string );

if ( date_greater_year_boolean() )
        return STATEMENT_GREATER_YEAR_MESSAGE;
else
        return NULL;

/* Usage */
enum statement_subclassification_option
    statement_resolve_subclassification_option(
        char *subclassification_option_string );

/* Process */

/* Usage */
enum statement_output_medium
    statement_resolve_output_medium(
        char *output_medium_string );

/* Process */

/* Usage */
char *statement_pdf_prompt(
    char *process_name );

/* Process */

/* Driver */
void statement_html_output(
    HTML_TABLE *html_table,
    char *secondary_title );

/* Process */
if ( secondary_title )
{
    printf( "<h1>%s</h1>\n", secondary_title );
}

void html_table_output(
    html_table,
    HTML_TABLE_ROWS_BETWEEN_HEADING );
```

statement_subclassification_option

statement_subclassification_aggregate
statement_subclassification_display
statement_subclassification_omit

statement_output_medium

statement_output_table
statement_output_PDF
statement_output_spreadsheet
statement_output_stdout

STATEMENT_CAPTION

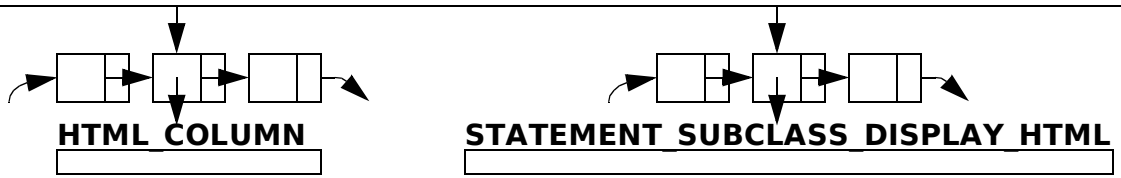
```
/* Usage */
STATEMENT_CAPTION *statement_caption_new(
    char *application_name,
    char *process_name,
    char *begin_date_string,
    char *end_date_time_string );

/* Process */
STATEMENT_CAPTION *statement_caption_calloc(
    void );

char *statement_caption_logo_filename(
    const char *STATEMENT_LOGO_FILENAME_KEY );

char *statement_caption_title(
    char *application_name,
    boolean (statement_caption_logo_filename())
       ? 1 : 0 /* exists_logo_filename */,
    char *process_name );

char *statement_caption_sub_title(
    begin_date_string,
    end_date_time_string );

char *statement_caption_combined(
    char *statement_caption_title(),
    char *statement_caption_sub_title() );

char *date_now16( date_utc_offset() );

char *statement_caption_frame_title(
    char *process_name,
    date_now16() );

/* Usage */
char *statement_caption_sub_title(
    char *begin_date_string,
    char *end_date_time_string );

/* Process */
if ( begin_date_string )
{
    char *begin_date_american =
       statement_date_american(
          begin_date_string );
}

if ( end_date_string )
{
    char *end_date_american(
       statement_date_american(
          end_date_time_string );
 }

if ( !begin_date_american && !end_date_american )
{
    *sub_title = '\0';
}
else
if ( !end_date_american )
{
    sprintf(
       static sub_title[],
       "Beginning: %s",
       begin_date_american );
}
else
if ( !begin_date_american )
{
    sprintf(
       static sub_title[],
       "Ending: %s",
       end_date_american );
}
else
{
    sprintf(
       static sub_title[],
       "Beginning: %s, Ending: %s",
       begin_date_american,
       end_date_american );
}

/* Attributes */
char *logo_filename;
char *title;
char *sub_title;
char *combined;
char *date_now16;
char *frame_title;
```

STATEMENT_DELTA

```
/* Usage */
STATEMENT_DELTA *statement_delta_new(
    double base_value,
    double change_value );

/* Process */
STATEMENT_DELTA *statement_delta_calloc(
    void );

double statement_delta_difference(
    double base_value,
    double change_value );

int statement_delta_percent(
    double base_value,
    double statement_delta_difference );

char *statement_delta_cell_string(
    double statement_delta_differencel(),
    int statement_delta_percent );

/* Attributes */
double difference;
int percent;
char *cell_string;
```

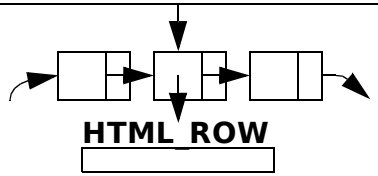
STATEMENT_SUBCLASS_DISPLAY_HTML_LIST

```
/* Usage */
STATEMENT_SUBCLASS_DISPLAY_HTML_LIST *
   statement_subclass_display_html_list_new(
      LIST *element_statement_list,
      LIST *statement_prior_year_list );

/* Process */
STATEMENT_SUBCLASS_DISPLAY_HTML_LIST *
   statement_subclass_display_html_list_calloc(
      void );

LIST *statement_subclass_display_html_list_column_list(
   statement_prior_year_list );

for ELEMENT *element in element_statement_list
{
   if ( !element->sum ) continue;

   list_set(
      list,
      statement_subclass_display_html_new(
         element,
         statement_prior_year_list ) );
}

/* Usage */
LIST *statement_subclass_display_html_list_column_list(
   LIST *statement_prior_year_list );

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

list_set(
   column_list,
   HTML_COLUMN *html_column_new(
      (char *)0,
      0 /* right_justify_boolean */ ) );

list_set(
   column_list,
   HTML_COLUMN *html_column_new(
      STATEMENT_ACCOUNT_HEADING,
      1 /* right_justified_boolean */ ) );

list_set(
   column_list,
   HTML_COLUMN *html_column_new(
      STATEMENT_SUBCLASSIFICATION_HEADING,
      1 /* right_justified_boolean */ ) );

list_set(
   column_list,
   HTML_COLUMN *html_column_new(
      STATEMENT_ELEMENT_HEADING,
      1 /* right_justified_boolean */ ) );

list_set(
   column_list,
   HTML_COLUMN *html_column_new(
      STATEMENT_STANDARDIZED_HEADING,
      1 /* right_justified_boolean */ ) );;

if ( list_length( statement_prior_year_list ) )
{
   LIST *statement_prior_year_heading_list(
      statement_prior_year_list );

   LIST *html_column_right_list(
      statement_prior_year_heading_list() );

   list_set_list(
      column_list,
      html_column_right_list() );
}

/* Usage */
LIST *statement_subclass_display_html_list_row_list(
   STATEMENT_SUBCLASS_DISPLAY_HTML_LIST *
      statement_subclass_display_html_list );

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

for STATEMENT_SUBCLASS_DISPLAY_HTML *
   statement_subclass_display_html in
      statement_subclass_display_html_list->list
{
   list_set_list(
      row_list,
      statement_subclass_display_html->row_list );
}

/* Attributes */
LIST *column_list;
LIST *list;
```



STATEMENT_SUBCLASS_DISPLAY_HTML (1)

```
/* Usage */
STATEMENT_SUBCLASS_DISPLAY_HTML *
    statement_subclass_display_html_new(
        ELEMENT *element,
        LIST *statement_prior_year_list );

/* Process */
STATEMENT_SUBCLASS_DISPLAY_HTML *
    statement_subclass_display_html_calloc(
        void );

list_set(
    row_list,
    HTML_ROW *statement_subclass_display_html_element_label_row(
        element->element_name,
        list_length( statement_prior_year_list )
            /* statement_prior_year_list_length */ ) );

for SUBCLASSIFICATION *subclassification in
    element->subclassification_statement_list
{
    if ( !subclassification->sum ) continue;

    boolean statement_subclassification_name_duplicated(
        element->element_name,
        subclassification->subclassification_name );

    if ( !statement_subclassification_name_duplicated() )
    {
        list_set(
            row_list,
            statement_subclass_display_html_subclassification_label_row(
                subclassiciation->subclassification_name,
                list_length( statement_prior_year_list )
                    /* statement_prior_year_list_length */ ) );
    }

    list_set_list(
        row_list,
        statement_subclass_display_html_account_row_list(
            subclassification->account_statement_list,
            statement_prior_year_list ) );

    if ( !statement_subclassification_name_duplicated() )
    {
        list_set(
            row_list,
            HTML_ROW *statement_subclass_display_html_subclassification_sum_row(
                subclassification->subclassification_name,
                subclassification->sum,
                subclassification->percent_of_asset,
                subclassification->percent_of_income,
                statement_prior_year_list ) );
    }
}

list_set(
    row_list,
    HTML_ROW *statement_subclass_display_html_element_sum_row(
        element->element_name,
        element->sum,
        percent_of_asset,
        percent_of_income,
        statement_prior_year_list ) );

/* Attribute */
LIST *row_list;
```



STATEMENT_SUBCLASS_DISPLAY_HTML (2)

```
/* Usage */
LIST *statement_subclass_display_html_account_row_list(
    LIST *account_statement_list,
    LIST *statement_prior_year_list );

/* Process */
for ACCOUNT *account in account_statement_list
{
    STATEMENT_ACCOUNT *statement_account =
        statement_account_new(
            (char *)0 /* transaction_date_time_closing */,
            0 /* element_accumulate_debit */,
            account->account_journal_latest,
            account->action_string,
            0 /* not round_dollar_boolean */,
            account );

    list_set(
        row_list,
        HTML_ROW *statement_subclass_display_html_account_row(
            statement_account,
            statement_prior_year_list ) );
}

/* Usage */
HTML_ROW *statement_subclass_display_html_element_label_row(
    char *element->element_name,
    int statement_prior_year_list_length );

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

char *statement_cell_label_datum( element_name );

list_set(
    cell_list,
    html_cell_new(
        statement_cell_label_datum(),
        1 /* large_boolean */,
        1 /* bold_boolean */ ) );

for ( i = 0;
      i < statement_prior_year_list_length + 4;
      i++ )
{
    list_set(
        cell_list,
        html_cell_new(
            (char *)0 /* datum */,
            0 /* not large_boolean */,
            0 /* not bold_boolean */ ) );
}

HTML_ROW *html_row_new( cell_list );

/* Usage */
HTML_ROW *statement_subclass_display_html_subclassification_label_row(
    char *subclassiciation->subclassification_name,
    int statement_prior_year_list_length );

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

char *statement_cell_label_datum( subclassification_name );

list_set(
    cell_list,
    html_cell_new(
        statement_cell_label_datum(),
        0 /* not large_boolean */,
        1 /* bold_boolean */ ) );

for ( i = 0;
      i < statement_prior_year_list_length + 4;
      i++ )
{
    list_set(
        cell_list,
        html_cell_new(
            (char *)0 /* datum */,
            0 /* not large_boolean */,
            0 /* not bold_boolean */ ) );
}

HTML_ROW *html_row_new( cell_list );
```

STATEMENT_SUBCLASS_DISPLAY_HTML (3)

```
/* Usage */
HTML_ROW *statement_subclass_display_html_subclassification_sum_row(
    char *subclassification->subclassification_name,
    double subclassification->sum,
    int subclassification->percent_of_asset,
    int subclassification->percent_of_income,
    LIST *statement_prior_year_list );

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

char *statement_cell_label_datum( subclassification_name );

list_set(
    cell_list,
    html_cell_new(
        statement_cell_label_datum(),
        0 /* not large_boolean */,
        1 /* bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        (char *)0 /* datum */,
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

char *statement_account_balance_string(
    sum /* balance */,
    (char *)0 /* account_action_string */,
    0 /* not round_dollar_boolean */ );

list_set(
    cell_list,
    html_cell_new(
        statement_account_balance_string(),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        (char *)0 /* datum */,
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

char *statement_account_percent_string(
    percent_of_asset,
    percent_of_income );

list_set(
    cell_list,
    html_cell_new(
        statement_account_percent_string(),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

if ( list_length( statement_prior_year_list ) )
{
    LIST *statement_prior_year_subclassification_data_list(
        subclassification_name,
        statement_prior_year_list );

    LIST *html_cell_list(
        statement_prior_year_subclassification_data_list() );

    list_set_list(
        cell_list,
        html_cell_list() );
}

HTML_ROW *html_row_new( cell_list );
```

STATEMENT_SUBCLASS_DISPLAY_HTML (4)

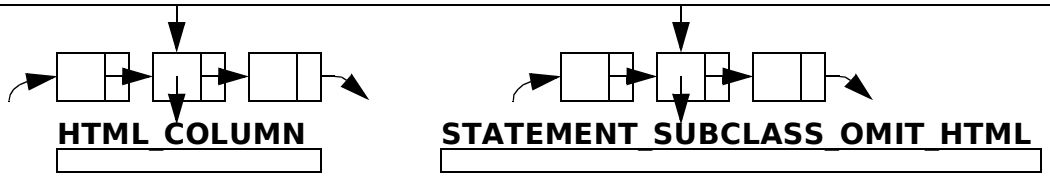
```
/* Usage */
HTML_ROW *statement_subclass_display_html_element_sum_row(
    char *element->element_name,
    double element->sum,
    int percent_of_asset,
    int percent_of_income,
    LIST *statement_prior_year_list );

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

char *statement_element_sum_name( element_name );

char *statement_cell_label_datum(
    statement_element_sum_name() );

list_set(
    cell_list,
    html_cell_new(
        statement_cell_label_datum(),
        1 /* large_boolean */,
        1 /* bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        (char *)0 /* datum */,
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        (char *)0 /* datum */,
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

char *statement_account_balance_string(
    sum /* balance */,
    (char *)0 /* account_action_string */,
    0 /* not round_dollar_boolean */ );

list_set(
    cell_list,
    html_cell_new(
        statement_account_balance_string(),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

char *statement_account_percent_string(
    percent_of_asset,
    percent_of_income );

list_set(
    cell_list,
    html_cell_new(
        statement_account_percent_string(),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

if ( list_length( statement_prior_year_list ) )
{
    LIST *statement_prior_year_element_data_list(
        element_name,
        statement_prior_year_list );

    LIST *html_cell_list(
        statement_prior_year_element_data_list() );

    list_set_list(
        cell_list,
        html_cell_list() );
}

HTML_ROW *html_row_new( cell_list );
```

STATEMENT_SUBCLASS_DISPLAY_HTML (5)

```
/* Usage */
HTML_ROW *
statement_subclass_display_html_account_row(
    STATEMENT_ACCOUNT *statement_account,
    LIST *statement_prior_year_list );

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

char *statement_cell_label_datum(
    char *statement_account->
        account->
        account_name );

list_set(
    cell_list,
    html_cell_new(
        statement_cell_label_datum(),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        statement_account->balance_string,
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        (char *)0 /* datum */,
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        (char *)0 /* datum */,
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        statement_account->percent_string,
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

if ( list_length( statement_prior_year_list ) )
{
    LIST *statement_prior_year_account_data_list(
        statement_account->
            account->
            account_name,
        statement_prior_year_list );

    LIST *html_cell_list(
        statement_prior_year_account_data_list() );

    list_set_list(
        cell_list,
        html_cell_list() );
}

HTML_ROW *html_row_new( cell_list );
```

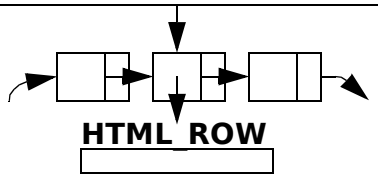
STATEMENT_SUBCLASS_OMIT_HTML_LIST

```
/* Usage */
STATEMENT_SUBCLASS_OMIT_HTML_LIST *
    statement_subclass_omit_html_list_new(
        LIST *element_statement_list,
        LIST *statement_prior_year_list );

/* Process */
STATEMENT_SUBCLASS_OMIT_HTML_LIST *
    statement_subclass_omit_html_list_calloc(
        void );

LIST *statement_subclass_omit_html_list_column_list(
    statement_prior_year_list );

LIST *list = list_new();

for ELEMENT *element in element_statement_list
{
    if ( !element->sum ) continue;

    list_set(
        list,
        statement_subclass_omit_html_new(
            element,
            statement_prior_year_list ) );
}

/* Usage */
LIST *statement_subclass_omit_html_list_column_list(
    LIST *statement_prior_year_list );

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

list_set(
    column_list,
    HTML_COLUMN *html_column_new(
        (char *)0 /* heading */,
        0 /* not right_justify_boolean */ ) );

list_set(
    column_list,
    HTML_COLUMN *html_column_new(
        STATEMENT_ACCOUNT_HEADING,
        1 /* right_justified_boolean */ ) );

list_set(
    column_list,
    HTML_COLUMN *html_column_new(
        STATEMENT_ELEMENT_HEADING,
        1 /* right_justified_boolean */ ) );

list_set(
    column_list,
    HTML_COLUMN *html_column_new(
        STATEMENT_STANDARDIZED_HEADING,
        1 /* right_justified_boolean */ ) );

if ( list_length( statement_prior_year_list ) )
{
    LIST *statement_prior_year_heading_list(
        statement_prior_year_list );

    LIST *html_column_right_list(
        statement_prior_year_heading_list() );

    list_set_list(
        column_list,
        html_column_right_list() );

/* Usage */
LIST *statement_subclass_omit_html_list_row_list(
    STATEMENT_SUBCLASS_OMIT_HTML_LIST *
        statement_subclass_omit_html_list );

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

for STATEMENT_SUBCLASS_OMIT_HTML *
    statement_subclass_omit_html in
        statement_subclass_omit_html_list->list
{
    list_set_list(
        row_list,
        statement_subclass_omit_html->row_list );
}

/* Attributes */
LIST *column_list;
LIST *list;
```



STATEMENT_SUBCLASS_OMIT_HTML (1)

```
/* Usage */
STATEMENT_SUBCLASS_OMIT_HTML *
    statement_subclass_omit_html_new(
        ELEMENT *element,
        LIST *statement_prior_year_list );

/* Process */
STATEMENT_SUBCLASS_OMIT_HTML *
    statement_subclass_omit_html_calloc(
        void );

LIST *row_list = list_new();

list_set(
    row_list,
    HTML_ROW *
    statement_subclass_omit_html_element_label_row(
        element->element_name,
        list_length( statement_prior_year_list )
            /* statement_prior_year_list_length */ ) );

for ACCOUNT *account in
    element->account_statement_list
{
    STATEMENT_ACCOUNT *statement_account =
        statement_account_new(
            (char *)0 /* transaction_date_time_closing */,
            0 /* element_accumulate_debit */,
            account->account_journal_latest,
            account->action_string,
            0 /* not round_dollar_boolean */,
            account );

    list_set(
        row_list,
        statement_subclass_omit_html_row(
            statement_account,
            statement_prior_year_list ) );
}

list_set(
    row_list,
    HTML_ROW *
    statement_subclass_omit_html_element_sum_row(
        element->element_name,
        element->sum,
        element->percent_of_asset,
        element->percent_of_income,
        statement_prior_year_list ) );

/* Usage */
HTML_ROW *statement_subclass_omit_html_row(
    STATEMENT_ACCOUNT *statement_account,
    LIST *statement_prior_year_list );

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

char *statement_cell_label_datum(
    statement_account->
        account->
        account_name );

list_set(
    cell_list,
    html_cell_new(
        statement_cell_label_datum(),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        statement_account->balance_string,
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        (char *)0 /* datum */,
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        statement_account->percent_string,
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

if ( list_length( statement_prior_year_list ) )
{
    LIST *statement_prior_year_account_data_list(
        statement_account->
            account->
            account_name,
        statement_prior_year_list );

    LIST *html_cell_list(
        statement_prior_year_account_data_list() );

    list_set_list(
        cell_list,
        html_cell_list() );
}

HTML_ROW *html_row_new( cell_list );

/* Attribute */
LIST *row_list;
```



STATEMENT_SUBCLASS_OMIT_HTML (2)

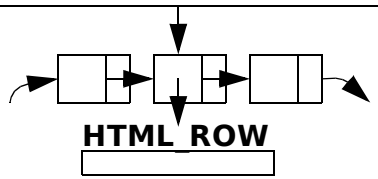
```
/* Usage */
HTML_ROW *
statement_subclass_omit_html_element_label_row(
    char *element->element_name,
    int statement_prior_year_list_length );

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

char *statement_cell_label_datum(
    element_name );

list_set(
    cell_list,
    html_cell_new(
        statement_cell_label_datum(),
        1 /* large_boolean */,
        1 /* bold_boolean */ ) );

for ( int i = 0;
        i < statement_prior_year_list_length + 2;
        i++ )
{
    list_set(
        cell_list,
        html_cell_new(
            (char *)0 /* datum */,
            0 /* not large_boolean */,
            0 /* not bold_boolean */ ) );
}

HTML_ROW *html_row_new( cell_list );

/* Usage */
HTML_ROW *
statement_subclass_omit_html_element_sum_row(
    char *element->element_name,
    double element->sum,
    int element->percent_of_asset,
    int element->percent_of_income,
    LIST *statement_prior_year_list );

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

char *statement_element_sum_name( element_name );

char *statement_cell_label_datum(
    statement_element_sum_name() );

list_set(
    cell_list,
    html_cell_new(
        statement_cell_label_datum(),
        1 /* large_boolean */,
        1 /* bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        (char *)0 /* datum */,
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

char *statement_account_balance_string(
    sum /* balance */,
    (char *)0 /* account_action_string */,
    0 /* not round_dollar_boolean */ );

list_set(
    cell_list,
    html_cell_new(
        statement_account_balance_string(),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

char *statement_account_percent_string(
    percent_of_asset,
    percent_of_income );

list_set(
    cell_list,
    html_cell_new(
        statement_account_percent_string(),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

if ( list_length( statement_prior_year_list ) )
{
    LIST *statement_prior_year_element_data_list(
        element_name,
        statement_prior_year_list );

    LIST *html_cell_list(
        statement_prior_year_element_data_list() );

    list_set_list(
        cell_list,
        html_cell_list() );
}

HTML_ROW *html_row_new( cell_list );
```

STATEMENT_SUBCLASS_AGGR_HTML_LIST

```
/* Usage */
STATEMENT_SUBCLASS_AGGR_HTML_LIST *
   statement_subclass_aggr_html_list_new(
      LIST *element_statement_list,
      LIST *statement_prior_year_list );

/* Process */
STATEMENT_SUBCLASS_AGGR_HTML_LIST *
   statement_subclass_aggr_html_list_calloc(
      void );

LIST *statement_subclass_aggr_html_list_column_list(
   statement_prior_year_list );

LIST *list = list_new();

for ELEMENT *element in element_statement_list
{
   if ( !element->sum ) continue;

   list_set(
      list,
      statement_subclass_aggr_html_new(
         element,
         statement_prior_year_list ) );
}

/* Usage */
LIST *statement_subclass_aggr_html_list_column_list(
   LIST *statement_prior_year_list );

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

list_set(
   column_list,
   HTML_COLUMN *html_column_new(
      (char *)0 /* heading */,
      0 /* not right_justify_boolean */ ) );

list_set(
   column_list,
   HTML_COLUMN *html_column_new(
      STATEMENT_BALANCE_HEADING,
      1 /* right_justified_boolean */ ) );

list_set(
   column_list,
   HTML_COLUMN *html_column_new(
      STATEMENT_ELEMENT_HEADING,
      1 /* right_justified_boolean */ ) );

list_set(
   column_list,
   HTML_COLUMN *html_column_new(
      STATEMENT_STANDARDIZED_HEADING,
      1 /* right_justified_boolean */ ) );

if ( list_length( statement_prior_year_list ) )
{
   LIST *statement_prior_year_heading_list(
      statement_prior_year_list );

   html_column_right_list(
      statement_prior_year_heading_list() );

   list_set_list(
      column_list,
      html_column_right_list() );
}

/* Usage */
LIST *statement_subclass_aggr_html_list_row_list(
   STATEMENT_SUBCLASS_AGGR_HTML_LIST *
      statement_subclass_aggr_html_list );

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

for STATEMENT_SUBCLASS_AGGR_HTML *
   statement_subclass_aggr_html in
      statement_subclass_aggr_html_list->list
{
   list_set_list(
      row_list,
      statement_subclass_aggr_html->row_list );
}

/* Attributes */
LIST *column_list;
LIST *list;
```

STATEMENT_SUBCLASS_AGGR_HTML (1)

```
/* Usage */
STATEMENT_SUBCLASS_AGGR_HTML *
    statement_subclass_aggr_html_new(
        ELEMENT *element,
        LIST *statement_prior_year_list );

/* Process */
STATEMENT_SUBCLASS_AGGR_HTML *
    statement_subclass_aggr_html_calloc(
        void );

list_set(
    row_list,
    HTML_ROW *
    statement_subclass_aggr_html_element_label_row(
        element->element_name,
        list_length( statement_prior_year_list )
            /* statement_prior_year_list_length */ ) );

for SUBCLASSIFICATION *subclassification in
    element->subclassification_statement_list
{
    if ( subclassification->sum )
    {
        list_set(
            row_list,
            statement_subclass_aggr_html_row(
                subclassification,
                statement_prior_year_list ) );
    }
}

list_set(
    row_list,
    HTML_ROW *
    statement_subclass_aggr_html_element_sum_row(
        element->element_name,
        element->sum,
        element->percent_of_asset,
        element->percent_of_income,
        statement_prior_year_list ) );

/* Usage */
HTML_ROW *statement_subclass_aggr_html_row(
    SUBCLASSIFICATION *subclassification,
    LIST *statement_prior_year_list );

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

char *statement_cell_label_datum(
    char *subclassification->subclassification_name );

list_set(
    cell_list,
    html_cell_new(
        statement_cell_label_datum(),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

char *statement_account_balance_string(
    subclassification->sum,
    (char *)0 /* account_action_string */,
    0 /* not round_dollar_boolean */ );

list_set(
    cell_list,
    html_cell_new(
        statement_account_balance_string(),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        (char *)0 /* datum */,
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

char *statement_account_percent_string(
    subclassification->percent_of_asset,
    subclassification->percent_of_income ),

list_set(
    cell_list,
    html_cell_new(
        statement_account_percent_string(),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

if ( list_length( statement_prior_year_list ) )
{
    LIST *
    statement_prior_year_subclassification_data_list(
        subclassification->subclassification_name,
        statement_prior_year_list );

    LIST *html_cell_list(
        statement_prior_year_subclassification_data_list() );

    list_set_list(
        cell_list,
        html_cell_list() );
}

HTML_ROW *html_row_new( cell_list );

/* Attribute */
LIST *row_list;
```

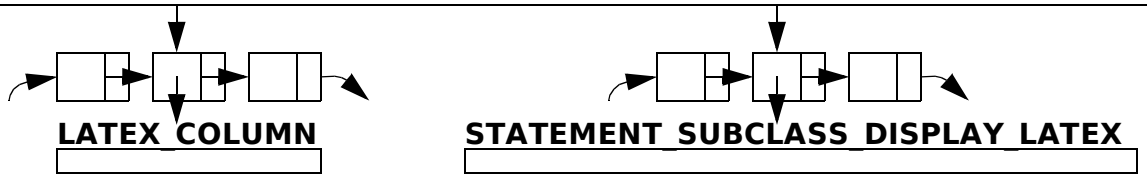


STATEMENT_SUBCLASS_AGGR_HTML (2)

```
/* Usage */
HTML_ROW *
statement_subclass_aggr_html_element_label_row(
    char *element->element_name,
    int statement_prior_year_list_length );

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

char *statement_cell_label_datum( element_name );

list_set(
    cell_list,
    html_cell_new(
        statement_cell_label_datum(),
        1 /* large_boolean */,
        1 /* bold_boolean */ ) );

for ( i = 0;
      i < statement_prior_year_list_length + 2;
      i++ )
{
    list_set(
        cell_list,
        html_cell_new(
            (char *)0 /* datum */,
            0 /* not large_boolean */,
            0 /* not bold_boolean */ ) );
}

HTML_ROW *html_row_new( cell_list );

/* Usage */
HTML_ROW *
statement_subclass_aggr_html_element_sum_row(
    char *element->element_name,
    double element->sum,
    int element->percent_of_asset,
    int element->percent_of_income,
    LIST *statement_prior_year_list );

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

char *statement_element_sum_name(
    element_name );

char *statement_cell_label_datum(
    statement_element_sum_name() );

list_set(
    cell_list,
    html_cell_new(
        statement_cell_label_datum(),
        1 /* large_boolean */,
        1 /* bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        (char *)0 /* datum */,
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

char *statement_account_balance_string(
    sum /* balance */,
    (char *)0 /* account_action_string */,
    0 /* not round_dollar_boolean */ );

list_set(
    cell_list,
    html_cell_new(
        statement_account_balance_string(),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

char *statement_account_percent_string(
    percent_of_asset,
    percent_of_income );

list_set(
    cell_list,
    html_cell_new(
        statement_account_percent(),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

if ( list_length( statement_prior_year_list ) )
{
    LIST *statement_prior_year_element_data_list(
        element_name,
        statement_prior_year_list );

    LIST *html_cell_list(
        statement_prior_year_element_data_list() );

    list_set_list(
        cell_list,
        html_cell_list() );
}

HTML_ROW *html_row_new( cell_list );
```

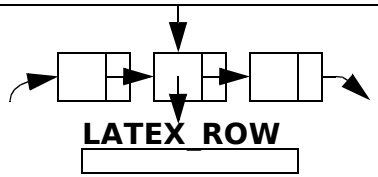
STATEMENT_SUBCLASS_DISPLAY_LATEX_LIST

```
/* Usage */
STATEMENT_SUBCLASS_DISPLAY_LATEX_LIST *
   statement_subclass_display_latex_list_new(
      LIST *element_statement_list,
      LIST *statement_prior_year_list );

/* Process */
STATEMENT_SUBCLASS_DISPLAY_LATEX_LIST *
   statement_subclass_display_latex_list_calloc(
      void );

LIST *statement_subclass_display_latex_list_column_list(
   statement_prior_year_list );

LIST *list = list_new();

for ELEMENT *element in element_statement_list
{
   if ( !element->sum ) continue;

   STATEMENT_SUBCLASS_DISPLAY_LATEX *
   statement_subclass_display_latex_new(
      element,
      statement_prior_year_list,
      statement_subclass_display_latex_list_column_list,
         /* latex_column_list */ );

   list_set(
      list,
      statement_subclass_display_latex_new() );
}

/* Usage */
LIST *statement_subclass_display_latex_list_column_list(
   LIST *statement_prior_year_list );

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

list_set(
   column_list,
   LATEX_COLUMN *latex_column_new(
      (char *)0 /* heading_string */,
      latex_column_text,
      0 /* float_decimal_count */,
      (char *)0 /* paragraph_size */,
      1 /* first_column_boolean */ ) );

list_set(
   column_list,
   LATEX_COLUMN *latex_column_new(
      STATEMENT_ACCOUNT_HEADING,
      latex_column_float,
      2 /* float_decimal_count */,
      (char *)0 /* paragraph_size */,
      0 /* not first_column_boolean */ ) );

list_set(
   column_list,
   LATEX_COLUMN *latex_column_new(
      STATEMENT_SUBCLASSIFICATION_HEADING,
      latex_column_float,
      2 /* float_decimal_count */,
      (char *)0 /* paragraph_size */,
      0 /* not first_column_boolean */ ) );

list_set(
   column_list,
   LATEX_COLUMN *latex_column_new(
      STATEMENT_ELEMENT_HEADING,
      latex_column_float,
      2 /* float_decimal_count */,
      (char *)0 /* paragraph_size */,
      0 /* not first_column_boolean */ ) );

LATEX_COLUMN *latex_column =
   latex_column_new(
      STATEMENT_STANDARDIZED_ESCAPED_HEADING,
      latex_column_text,
      0 /* float_decimal_count */,
      (char *)0 /* paragraph_size */,
      0 /* not first_column_boolean */ ) );

latex_column->right_justify_boolean = 1;

list_set( column_list, latex_column );

if ( list_length( statement_prior_year_list ) )
{
   LIST *statement_prior_year_heading_list(
      statement_prior_year_list );

   LIST *latex_column_text_list(
      statement_prior_year_heading_list(),
      0 /* not first_column_boolean */,
      1 /* right_justify_boolean */ );

   list_set_list(
      column_list,
      latex_column_text_list() );
}

/* Usage */
LIST *statement_subclass_display_latex_list_row_list(
   STATEMENT_SUBCLASS_DISPLAY_LATEX_LIST *
      statement_subclass_display_latex_list );

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

for STATEMENT_SUBCLASS_DISPLAY_LATEX *
   statement_subclass_display_latex in
      statement_subclass_display_latex_list->list
{
   list_set_list(
      row_list,
      statement_subclass_display_latex->row_list );
}

/* Attributes */
LIST *column_list;
LIST *list;
```



STATEMENT_SUBCLASS_DISPLAY_LATEX (1)

```
/* Usage */
STATEMENT_SUBCLASS_DISPLAY_LATEX *
    statement_subclass_display_latex_new(
        ELEMENT *element,
        LIST *statement_prior_year_list,
        LIST *latex_column_list );

/* Process */
STATEMENT_SUBCLASS_DISPLAY_LATEX *
    statement_subclass_display_latex_calloc(
        void );

LIST *row_list = list_new();

list_set(
    row_list,
    LATEX_ROW *statement_subclass_display_latex_element_label_row(
        element->element_name,
        list_length( statement_prior_year_list )
            /* statement_prior_year_list_length */,
        latex_column_list );

for SUBCLASSIFICATION *subclassification in
    element->subclassification_statement_list
{
    if ( !subclassification->sum ) continue;

    boolean statement_subclassification_name_duplicated(
        element->element_name,
        subclassification->subclassification_name );

    if ( !statement_subclassification_name_duplicated() )
    {
        list_set(
            row_list,
            LATEX_ROW *statement_subclass_display_latex_subclassification_label_row(
                subclassification->subclassification_name,
                list_length( statement_prior_year_list )
                    /* statement_prior_year_list_length */,
                latex_column_list );
    }

    list_set_list(
        row_list,
        statement_subclass_display_latex_account_row_list(
            subclassification->account_statement_list,
            statement_prior_year_list,
            latex_column_list ) );

    if ( !statement_subclassification_name_duplicated() )
    {
        list_set(
            row_list,
            LATEX_ROW *statement_subclass_display_latex_subclassification_sum_row(
                subclassification->subclassification_name,
                subclassification->sum,
                subclassification->percent_of_asset,
                subclassification->percent_of_income,
                statement_prior_year_list,
                latex_column_list );
    }
}

list_set(
    row_list,
    LATEX_ROW *statement_subclass_display_latex_element_sum_row(
        element->element_name,
        element->sum,
        percent_of_asset,
        percent_of_income,
        statement_prior_year_list,
        latex_column_list );

/* Attribute */
LIST *row_list;
```



STATEMENT_SUBCLASS_DISPLAY_LATEX (2)

```
/* Usage */
LATEX_ROW *statement_subclass_display_latex_element_label_row(
    char *element->element_name,
    int statement_prior_year_list_length,
    LIST *latex_column_list );

/* Process */
boolean list_rewind( latex_column_list );

LIST *cell_list = list_new();

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

char *statement_cell_label_datum(
    element_name );

list_set(
    cell_list,
    latex_cell_new(
       latex_column,
       0 /* not first_row_boolean */,
       statement_cell_label_datum(),
       1 /* large_boolean */,
       1 /* bold_boolean */ ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

boolean list_next( latex_column_list );

for ( i = 0;
       i < statement_prior_year_list_length + 4;
       i++ )
{
    list_set(
       cell_list,
       latex_cell_small_new(
          latex_column,
          0 /* not first_row_boolean */,
          (char *)0 /* datum */ ) );

    LATEX_COLUMN *latex_column =
       list_get( latex_column_list );

    boolean list_next( latex_column_list );
}

LATEX_ROW *latex_row_new( cell_list );
```

STATEMENT_SUBCLASS_DISPLAY_LATEX (3)

```
/* Usage */
LATEX_ROW *statement_subclass_display_latex_subclassification_sum_row(
    char *subclassification->subclassification_name,
    double subclassification->sum,
    int subclassification->percent_of_asset,
    int subclassification->percent_of_income,
    LIST *statement_prior_year_list,
    LIST *latex_column_list );

/* Process */
boolean list_rewind( latex_column_list );

LIST *cell_list = list_new();

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
boolean list_next( latex_column_list );

char *statement_cell_label_datum(
    subclassification_name );

list_set(
    cell_list,
    latex_cell_new(
        latex_column,
        0 /* not first_row_boolean */,
        statement_cell_label_datum(),
        0 /* not large_boolean */,
        1 /* bold_boolean */ ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
boolean list_next( latex_column_list );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        (char *)0 /* datum */ ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
list_next( latex_column_list );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        (char *)0 /* datum */ ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
list_next( latex_column_list );

char *statement_account_balance_string(
    sum /* balance */,
    (char *)0 /* account_action_string */,
    0 /* not round_dollar_boolean */ );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        statement_account_balance_string() ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
list_next( latex_column_list );

char *statement_account_percent_string(
    percent_of_asset,
    percent_of_income );

void list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        statement_account_percent_string() ) );

if ( list_length( statement_prior_year_list ) )
{
    LIST *statement_prior_year_subclassification_data_list(
        subclassifiction_name,
        statement_prior_year_list );

    LIST *latex_cell_list(
        __FUNCTION__,
        latex_column_list,
        statement_prior_year_subclassification_data_list() );

    void list_set_list(
        cell_list,
        latex_cell_list() );
}

LATEX_ROW *latex_row_new( cell_list );
```

STATEMENT_SUBCLASS_DISPLAY_LATEX (4)

```
/* Usage */
LATEX_ROW *statement_subclass_display_latex_element_sum_row(
    char *element->element_name,
    double element->sum,
    int element->percent_of_asset,
    int element->percent_of_income,
    LIST *statement_prior_year_list,
    LIST *latex_column_list );

/* Process */
boolean list_rewind( latex_column_list );
LIST *cell_list = list_new();

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
boolean list_next( latex_column_list );

char *statement_element_sum_name(
    element_name );

char *statement_cell_label_datum(
    statement_element_sum_name() );

list_set(
    cell_list,
    latex_cell_new(
        latex_column,
        0 /* not first_row_boolean */,
        statement_cell_label_datum(),
        1 /* large_boolean */,
        1 /* bold_boolean */ ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
boolean list_next( latex_column_list );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        (char *)0 /* datum */ ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
list_next( latex_column_list );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        (char *)0 /* datum */ ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
list_next( latex_column_list );

char *statement_account_balance_string(
    sum /* balance */,
    (char *)0 /* account_action_string */,
    0 /* not round_dollar_boolean */ );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        statement_account_balance_string() ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
list_next( latex_column_list );

char *statement_account_percent_string(
    percent_of_asset,
    percent_of_income );

void list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        statement_account_percent_string() ) );

if ( list_length( statement_prior_year_list ) )
{
    LIST *statement_prior_year_element_data_list(
        element_name,
        statement_prior_year_list );

    LIST *latex_cell_list(
        __FUNCTION__,
        latex_column_list,
        statement_prior_year_element_data_list() );

    void list_set_list(
        cell_list,
        latex_cell_list() );
}

LATEX_ROW *latex_row_new( cell_list );
```

STATEMENT_SUBCLASS_DISPLAY_LATEX (5)

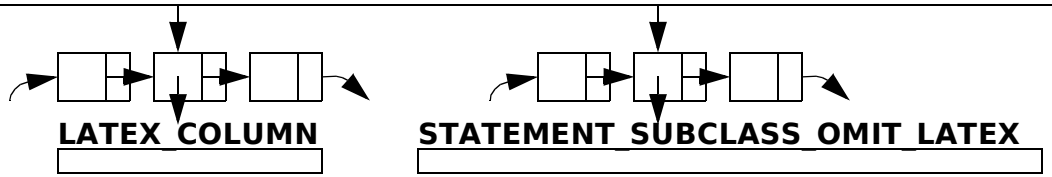
```
/* Usage */
LATEX_ROW *statement_subclass_display_latex_subclassification_label_row(
    char *subclassification_name,
    int statement_prior_year_list_length,
    LIST *latex_column_list );

/* Process */
boolean list_rewind( latex_column_list );

LIST *cell_list = list_new();

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

char *statement_cell_label_datum(
    subclassification_name );

list_set(
    cell_list,
    latex_cell_new(
       latex_column,
       0 /* not first_row_boolean */,
       statement_cell_label_datum(),
       0 /* not large_boolean */,
       1 /* bold_boolean */ ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

boolean list_next( latex_column_list );

for ( i = 0;
       i < statement_prior_year_list_length + 4;
       i++ )
{
    list_set(
       cell_list,
       latex_cell_small_new(
          latex_column,
          (char *)0 /* datum */ ) );

    LATEX_COLUMN *latex_column =
       list_get( latex_column_list );

    boolean list_next( latex_column_list );
}

LATEX_ROW *latex_row_new( cell_list );
```

STATEMENT_SUBCLASS_DISPLAY_LATEX (6)

```
/* Usage */
LIST *
statement_subclass_display_latex_account_row_list(
    LIST *account_statement_list,
    LIST *statement_prior_year_list,
    LIST *latex_column_list );

/* Process */
for ACCOUNT *account in account_statement_list
{
    STATEMENT_ACCOUNT *statement_account =
        statement_account_new(
            (char *)0 /* end_date_time_string */,
            0 /* element_accumulate_debit */,
            account->account_journal_latest,
            (char *)0 /* account_action_string */,
            0 /* not round_dollar_boolean */,
            account );

    LATEX_ROW *
    statement_subclass_display_latex_account_row(
        statement_account,
        statement_prior_year_list,
        latex_column_list );

    list_set(
        row_list,
        statement_subclass_display_latex_row() );
}

/* Usage */
LATEX_ROW *
statement_subclass_display_latex_account_row(
    STATEMENT_ACCOUNT *statement_account,
    LIST *statement_prior_year_list,
    LIST *latex_column_list );

/* Process */
boolean list_rewind( latex_column_list );

LIST *cell_list = list_new();

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

char *statement_cell_label_datum(
    char *statement_account->
        account->
        account_name );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        statement_cell_label_datum() ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        statement_account->balance_string ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        (char *)0 /* datum */ ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        (char *)0 /* datum */ ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        statement_account->percent_string ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

if ( list_length( statement_prior_year_list ) )
{
    LIST *statement_prior_year_account_data_list(
        statement_account->
            account->
            account_name,
        statement_prior_year_list );

    LIST *latex_cell_list(
        __FUNCTION__,
        latex_column_list,
        statement_prior_year_account_data_list() );

    void list_set_list(
        cell_list,
        latex_cell_list() );
}

LATEX_ROW *latex_row_new( cell_list );
```

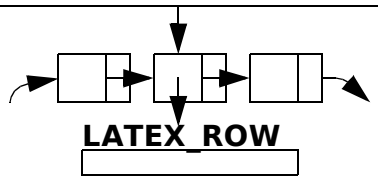
STATEMENT_SUBCLASS_OMIT_LATEX_LIST

```
/* Usage */
STATEMENT_SUBCLASS_OMIT_LATEX_LIST *
    statement_subclass_omit_latex_list_new(
        LIST *element_statement_list,
        LIST *statement_prior_year_list );

/* Process */
STATEMENT_SUBCLASS_OMIT_LATEX_LIST *
    statement_subclass_omit_latex_list_calloc(
        void );

LIST *statement_subclass_omit_latex_list_column_list(
    statement_prior_year_list );

LIST *list = list_new();

for ELEMENT *element in element_statement_list
{
    if ( !element->sum ) continue;

    list_set(
        list,
        STATEMENT_SUBCLASS_OMIT_LATEX *
        statement_subclass_omit_latex_new(
            element,
            statement_prior_year_list,
            statement_subclass_omit_latex_list_column_list()
                /* latex_column_list */ ) );
}

/* Usage */
LIST *statement_subclass_omit_latex_list_column_list(
    LIST *statement_prior_year_list );

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

list_set(
    column_list,
    latex_column_new(
        (char *)0 /* heading_string */,
        latex_column_text,
        0 /* float_decimal_count */,
        (char *)0 /* paragraph_size */,
        1 /* first_column_boolean */ ) );

list_set(
    column_list,
    latex_column_new(
        STATEMENT_ACCOUNT_HEADING,
        latex_column_float,
        2 /* float_decimal_count */,
        (char *)0 /* paragraph_size */,
        0 /* not first_column_boolean */ ) );

list_set(
    column_list,
    latex_column_new(
        STATEMENT_ELEMENT_HEADING,
        latex_column_float,
        2 /* float_decimal_count */,
        (char *)0 /* paragraph_size */,
        0 /* not first_column_boolean */ ) );

LATEX_COLUMN *latex_column =
    latex_column_new(
        STATEMENT_STANDARDIZED_ESCAPED_HEADING,
        latex_column_text,
        0 /* float_decimal_count */,
        (char *)0 /* paragraph_size */,
        0 /* not first_column_boolean */ ) );

latex_column->right_justify_boolean = 1;

list_set( column_list, latex_column );

if ( list_length( statement_prior_year_list ) )
{
    LIST *statement_prior_year_heading_list(
        statement_prior_year_list );

    LIST *latex_column_text_list(
        statement_prior_year_heading_list(),
        0 /* not first_column_boolean */,
        1 /* right_justify_boolean */ );

    list_set_list(
        column_list,
        latex_column_text_list() );
}

/* Usage */
LIST *statement_subclass_omit_latex_list_row_list(
    STATEMENT_SUBCLASS_OMIT_LATEX_LIST *
        statement_subclass_omit_latex_list );

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

for STATEMENT_SUBCLASS_OMIT_LATEX *
    statement_subclass_omit_latex in
        statement_subclass_omit_latex_list->list
{
    list_set_list(
        row_list,
        statement_subclass_omit_latex->row_list );
}

/* Attributes */
LIST *column_list;
LIST *list;
```



STATEMENT_SUBCLASS_OMIT_LATEX (1)

```
/* Usage */
STATEMENT_SUBCLASS_OMIT_LATEX *
    statement_subclass_omit_latex_new(
        ELEMENT *element,
        LIST *statement_prior_year_list,
        LIST *latex_column_list );

/* Process */
STATEMENT_SUBCLASS_OMIT_LATEX *
    statement_subclass_omit_latex_calloc(
        void );

LIST *row_list = list_new();

list_set(
    row_list,
    LATEX_ROW *
    statement_subclass_omit_latex_element_label_row(
        element->element_name,
        list_length( statement_prior_year_list )
            /* statement_prior_year_list_length */ ),
        latex_column_list );

for ACCOUNT *account in
    element->account_statement_list
{
    STATEMENT_ACCOUNT *statement_account =
        statement_account_new(
            (char *)0 /* transaction_date_time_closing */,
            0 /* element_accumulate_debit */,
            account->account_journal_latest,
            (char *)0 /* account_action_string */,
            0 /* not round_dollar_boolean */,
            account );

    list_set(
        row_list,
        statement_subclass_omit_latex_row(
            statement_account,
            statement_prior_year_list,
            latex_column_list ) );
}

list_set(
    row_list,
    LATEX_ROW *
    statement_subclass_omit_latex_element_sum_row(
        element->element_name,
        element->sum,
        element->percent_of_asset,
        element->percent_of_income,
        statement_prior_year_list,
        latex_column_list ) );

/* Attributes */
LIST *row_list;
```



STATEMENT_SUBCLASS_OMIT_LATEX (2)

```
/* Usage */
LATEX_ROW *
statement_subclass_omit_latex_element_label_row(
    char *element->element_name,
    int statement_prior_year_list_length,
    LIST *latex_column_list );

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

list_rewind( latex_column_list );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

char *statement_cell_label_datum(
    element_name );

list_set(
    cell_list,
    latex_cell_new(
        latex_column,
        0 /* not first_row_boolean */,
        statement_cell_label_datum(),
        1 /* large_boolean */,
        1 /* bold_boolean */ ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        (char *)0 /* datum */ ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        (char *)0 /* datum */ ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        (char *)0 /* datum */ ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

while ( statement_prior_year_list_length-- )
{
    list_set(
        cell_list,
        latex_cell_small_new(
            latex_column,
            0 /* not first_row_boolean */,
            (char *)0 /* datum */ ) );

    LATEX_COLUMN *latex_column =
        list_get( latex_column_list );

    list_next( latex_column_list );
}

LATEX_ROW *latex_row_new( cell_list );
```

STATEMENT_SUBCLASS_OMIT_LATEX (3)

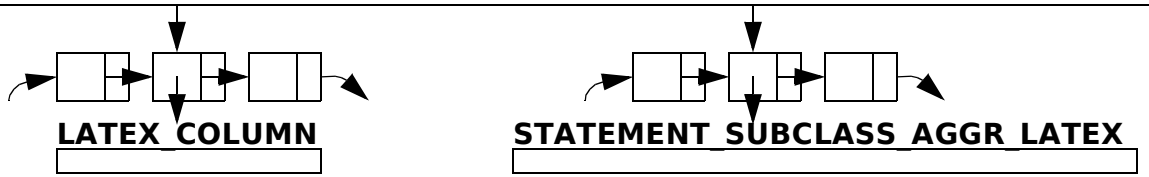
```
/* Usage */
LATEX_ROW *
statement_subclass_omit_latex_element_sum_row(
    char *element->element_name,
    double element->sum,
    int element->percent_of_asset,
    int element->percent_of_income,
    LIST *statement_prior_year_list,
    LIST *latex_column_list );

/* Process */
boolean list_rewind( latex_column_list );

LIST *cell_list = list_new();

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

boolean list_next( latex_column_list );

char *statement_element_sum_name(
    element_name );

char *statement_cell_label_datum(
    statement_element_sum_name() );

list_set(
    cell_list,
    latex_cell_new(
        latex_column,
        0 /* not first_row_boolean */,
        statement_cell_label_datum(),
        1 /* large_boolean */,
        1 /* bold_boolean */ ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

boolean list_next( latex_column_list );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        (char *)0 /* datum */ ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

boolean list_next( latex_column_list );

char *statement_account_balance_string(
    sum /* balance */,
    (char *)0 /* account_action_string */,
    0 /* not round_dollar_boolean * );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        statement_account_balance_string() ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

char *statement_account_percent_string(
    percent_of_asset,
    percent_of_income );

void list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        statement_account_percent_string() );

if ( list_length( statement_prior_year_list ) )
{
    LIST *statement_prior_year_element_data_list(
        element_name,
        statement_prior_year_list );

    LIST *latex_cell_list(
        __FUNCTION__,
        latex_column_list,
        statement_prior_year_element_data_list() );

    void list_set_list(
        cell_list,
        latex_cell_list() );
}

LATEX_ROW *latex_row_new( cell_list );
```

STATEMENT_SUBCLASS_OMIT_LATEX (4)

```
/* Usage */
LATEX_ROW *
statement_subclass_omit_latex_row(
    STATEMENT_ACCOUNT *statement_account,
    LIST *statement_prior_year_list,
    LIST *latex_column_list );

/* Process */
boolean list_rewind( latex_column_list );

LIST *cell_list = list_new();

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

boolean list_next( latex_column_list );

char *statement_cell_label_datum(
    char *statement_account->
        account->
        account_name );

void list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        statement_cell_label_datum() );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

boolean list_next( latex_column_list );

void list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        statement_account->balance_string ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

boolean list_next( latex_column_list );

void list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        (char *)0 /* datum */ ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

boolean list_next( latex_column_list );

void list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        statement_account->percent_string ) );

if ( list_length( statement_prior_year_list ) )
{
    LIST *statement_prior_year_account_data_list(
        statement_account->
            account->
            account_name,
        statement_prior_year_list );

    LIST *latex_cell_list(
        __FUNCTION__,
        latex_column_list,
        statement_prior_year_account_data_list() );

    void list_set_list(
        cell_list,
        latex_cell_list() );
}

LATEX_ROW *latex_row_new( cell_list );
```

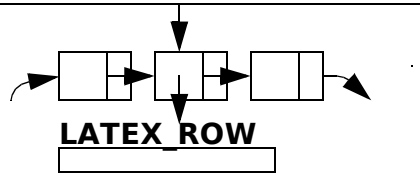
STATEMENT_SUBCLASS_AGGR_LATEX_LIST

```
/* Usage */
STATEMENT_SUBCLASS_AGGR_LATEX_LIST *
    statement_subclass_aggr_latex_list_new(
        LIST *element_statement_list,
        LIST *statement_prior_year_list );

/* Process */
STATEMENT_SUBCLASS_AGGR_LATEX_LIST *
    statement_subclass_aggr_latex_list_calloc(
        void );

LIST *statement_subclass_aggr_latex_list_column_list(
    statement_prior_year_list );

LIST *list = list_new();

for ELEMENT *element in element_statement_list
{
    if ( !element->sum ) continue;

    list_set(
        list,
        statement_subclass_aggr_latex_new(
            element,
            statement_prior_year_list,
            statement_subclass_aggr_latex_list_column_list
                /* latex_column_list */ ) );
}

/* Usage */
LIST *statement_subclass_aggr_latex_list_column_list(
    LIST *statement_prior_year_list );

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

list_set(
    column_list,
    latex_column_new(
        (char *)0 /* heading_string */,
        latex_column_text,
        0 /* float_decimal_count */,
        (char *)0 /* paragraph_size */,
        1 /* first_column_boolean */ ) );

list_set(
    column_list,
    latex_column_new(
        STATEMENT_BALANCE_HEADING,
        latex_column_float,
        2 /* float_decimal_count */,
        (char *)0 /* paragraph_size */,
        0 /* not first_column_boolean */ ) );

list_set(
    column_list,
    latex_column_new(
        STATEMENT_ELEMENT_HEADING,
        latex_column_float,
        2 /* float_decimal_count */,
        (char *)0 /* paragraph_size */,
        0 /* not first_column_boolean */ ) );

LATEX_COLUMN *latex_column =
    latex_column_new(
        STATEMENT_STANDARDIZED_ESCAPED_HEADING,
        latex_column_text,
        0 /* float_decimal_count */,
        (char *)0 /* paragraph_size */,
        0 /* not first_column_boolean */ ) );

latex_column->right_justify_boolean = 1;

list_set( column_list, latex_column );

if ( list_length( statement_prior_year_list ) )
{
    LIST *statement_prior_year_heading_list(
        statement_prior_year_list );

    LIST *latex_column_text_list(
        statement_prior_year_heading_list(),
        0 /* not first_column_boolean */,
        1 /* right_justify_boolean */ );

    list_set_list(
        column_list,
        latex_column_text_list() );
}

/* Usage */
LIST *statement_subclass_aggr_latex_list_row_list(
    STATEMENT_SUBCLASS_AGGR_LATEX_LIST *
        statement_subclass_aggr_latex_list );

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

for STATEMENT_SUBCLASS_AGGR_LATEX *
    statement_subclass_aggr_latex in
        statement_subclass_aggr_latex_list->list
{
    list_set_list(
        row_list,
        statement_subclass_aggr_latex->row_list );
}

/* Attributes */
LIST *column_list;
LIST *list;
```



STATEMENT_SUBCLASS_AGGR_LATEX (1)

```
/* Usage */
STATEMENT_SUBCLASS_AGGR_LATEX *
    statement_subclass_aggr_latex_new(
        ELEMENT *element,
        LIST *statement_prior_year_list,
        LIST *latex_column_list );

/* Process */
STATEMENT_SUBCLASS_AGGR_LATEX *
    statement_subclass_aggr_latex_calloc(
        void );

list_set(
    row_list,
    LATEX_ROW *
    statement_subclass_aggr_latex_element_label_row(
        char *element->element_name,
        int list_length( statement_prior_year_list )
            /* statement_prior_year_list_length */ ),
        LIST *latex_column_list );

for SUBCLASSIFICATION *subclassification in
    element->subclassification_statement_list
{
    if ( !subclassification->sum ) continue;

    list_set(
        row_list,
        statement_subclass_aggr_latex_row(
            subclassification,
            statement_prior_year_list,
            latex_column_list ) );
}

list_set(
    row_list,
    LATEX_ROW *
    statement_subclass_aggr_latex_element_sum_row(
        element->element_name,
        element->sum,
        element->percent_of_asset,
        element->percent_of_income,
        statement_prior_year_list,
        latex_column_list ) );

/* Attribute */
LIST *row_list;
```



STATEMENT_SUBCLASS_AGGR_LATEX (2)

```
/* Usage */
LATEX_ROW *
statement_subclass_aggr_latex_element_label_row(
    char *element_name,
    int statement_prior_year_list_length,
    LIST *latex_column_list );

/* Process */
list_rewind( latex_column_list );

LIST *cell_list = list_new();

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

char *statement_cell_label_datum(
    element_name );

void list_set(
    cell_list,
    latex_cell_new(
        latex_column,
        0 /* not first_row_boolean */,
        statement_cell_label_datum(),
        1 /* large_boolean */,
        1 /* bold_boolean */ ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

for ( i = 0;
      i < statement_prior_year_list_length + 3;
      i++ )
{
    list_set(
        cell_list,
        latex_cell_small_new(
            latex_column,
            0 /* not first_row_boolean */,
            (char *)0 /* datum */ ) );

    LATEX_COLUMN *latex_column =
        list_get( latex_column_list );

    list_next( latex_column_list );
}

LATEX_ROW *latex_row_new( cell_list );
```

STATEMENT_SUBCLASS_AGGR_LATEX (3)

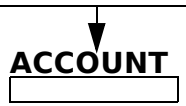
```
//* Usage */
LATEX_ROW *
statement_subclass_aggr_latex_element_sum_row(
    char *element->element_name,
    double element->sum,
    int element->percent_of_asset,
    int element->percent_of_income,
    LIST *statement_prior_year_list,
    LIST *latex_column_list );

/* Process */
list_rewind( latex_column_list );

LIST *cell_list = list_new();

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

char *statement_element_sum_name(
    element_name );

char *statement_cell_label_datum(
    statement_element_sum_name() );

void list_set(
    cell_list,
    latex_cell_new(
        latex_column,
        0 /* not first_row_boolean */,
        statement_cell_label_datum(),
        1 /* large_boolean */,
        1 /* bold_boolean */ ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        (char *)0 /* datum */ ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

char *statement_account_balance_string(
    sum /* balance */,
    (char *)0 /* account_action_string */,
    0 /* not round_dollar_boolean */ );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        statement_account_balance_string() );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

char *statement_account_percent_string(
    percent_of_asset,
    percent_of_income );

void list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        statement_account_percent_string() );

if ( list_length( statement_prior_year_list ) )
{
    LIST *statement_prior_year_element_data_list(
        element_name,
        statement_prior_year_list );

    LIST *latex_cell_list(
        __FUNCTION__,
        latex_column_list,
        statement_prior_year_element_data_list() );

    void list_set_list(
        cell_list,
        latex_cell_list() );
}

LATEX_ROW *latex_row_new( cell_list );
```

STATEMENT_SUBCLASS_AGGR_LATEX (4)

```
/* Usage */
LATEX_ROW *statement_subclass_aggr_latex_row(
    SUBCLASSIFICATION *subclassification,
    LIST *statement_prior_year_list,
    LIST *latex_column_list );

/* Process */
list_rewind( latex_column_list );

LIST *cell_list = list_new();

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

char *statement_cell_label_datum(
    subclassification->subclassification_name );

void list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        statement_cell_label_datum() ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

char *statement_account_balance_string(
    subclassification->sum,
    (char *)0 /* account_action_string */,
    0 /* not round_dollar_boolean */ );

void list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        statement_account_balance_string() );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

void list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        (char *)0 ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );

list_next( latex_column_list );

char *statement_account_percent_string(
    subclassification->percent_of_asset,
    subclassification->percent_of_income ),

void list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        statement_account_percent_string() ) );

if ( list_length( statement_prior_year_list ) )
{
    LIST *
    statement_prior_year_subclassification_data_list(
        subclassifiction_name,
        statement_prior_year_list );

    LIST *latex_cell_list(
        __FUNCTION__,
        latex_column_list,
        statement_prior_year_subclassification_data_list() );

    void list_set_list(
        cell_list,
        latex_cell_list() );
}

LATEX_ROW *latex_row_new( cell_list );
```

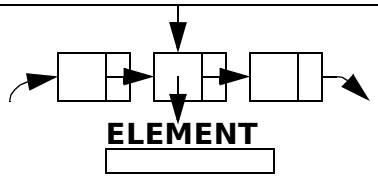
STATEMENT_ACCOUNT

```
/* Usage */
STATEMENT_ACCOUNT *statement_account_new(
    char *end_date_time_string,
    boolean element_accumulate_debit,
    ACCOUNT_JOURNAL *account_journal_latest,
    char *account_action_string,
    boolean round_dollar_boolean,
    ACCOUNT *account );

/* Process */
STATEMENT_ACCOUNT *statement_account_calloc(
    void );

double balance = account_journal_latest->balance;

if ( balance < 0.0 )
{
    balance = -balance;
    element_accumulate_debit =
       1 - element_accumulate_debit;
}

char *statement_account_balance_string(
    balance,
    account_action_string,
    round_dollar_boolean );

if ( element_accumulate_debit )
{
    debit_string =
       statement_account_balance_string();
}
else
{
    credit_string =
       statement_account_balance_string();
}

char *statement_account_asset_percent_string(
    int account->percent_of_asset );

char *statement_account_revenue_percent_string(
    int account->percent_of_income );

char *statement_account_percent_string(
    int account->percent_of_asset,
    int account->percent_of_income );

if ( transaction_date_time_closing )
{
    int date_days_between(
       account_journal_latest->
          transaction_date_time,
       end_date_time_string );

    boolean within_days_between_boolean =
       (date_days_between() <=
       STATEMENT_DAYS_FOR_EMPHASIS);
}

/* Usage */
char *statement_account_balance_string(
    double balance,
    char *account_action_string,
    boolean round_dollar_boolean );

/* Process */
char *statement_account_money_string(
    double balance,
    boolean round_dollar_boolean );

if ( account_action_string )
{
    sprintf(
       balance_string[],
       "<a href=\"%s\">%s</a>",
       account_action_string,
       statement_account_money_string() );
}
else
{
    strcpy(
       balance_string[],
       statement_account_money_string() );
}

char *strdup( balance_string );

/* Attributes */
char *balance_string;
char *debit_string;
char *credit_string;
char *asset_percent_string;
char *revenue_percent_string;
char *percent_string;
ACCOUNT *account;
```



STATEMENT_PRIOR_YEAR (1)

```
/* Usage */
LIST *statement_prior_year_list(
    LIST *element_name_list,
    char *end_date_time_string,
    int prior_year_count,
    STATEMENT *statement );

/* Process */
for years_ago = 1 to prior_year_count
{
    list_set(
       prior_year_list,
       STATEMENT_PRIOR_YEAR *
          statement_prior_year_fetch(
             element_name_list,
             end_date_time_string,
             years_ago,
             statement->
                element_statement_list ) );
}

/* Usage */
STATEMENT_PRIOR_YEAR *
    statement_prior_year_fetch(
       LIST *element_name_list,
       char *end_date_time_string,
       int years_ago,
       LIST *current_element_statement_list );

/* Process */
STATEMENT_PRIOR_YEAR *
    statement_prior_year_calloc(
       void );

char *statement_prior_year_date_time_string(
    end_date_time_string,
    years_ago );

LIST *element_statement_list(
    element_name_list,
    statement_prior_year_date_time_string(),
    1 /* fetch_subclassification_list */,
    1 /* fetch_account_list */,
    1 /* fetch_journal_latest *,
    0 /* not fetch_transaction */ );

void element_list_sum_set(
    element_statement_list() );

for ELEMENT *current_element in
    current_element_statement_list
{
    ELEMENT *prior_element =
       element_seek(
          current_element->element_name,
          element_statement_list() );

    if ( prior_element )
    {
       void element_delta_prior_percent_set(
          prior_element /* in/out */,
          current_element );
    }
}

/* Usage */
char *statement_prior_year_date_time_string(
    char *end_date_time_string,
    int years_ago );

/* Process */
char *date_time_string =
    date_display19(
       date_subtract_year(
          date_19new(
             end_date_time_string ),
          years_ago ) );

/* Usage */
LIST *statement_prior_year_account_data_list(
    char *account_name,
    LIST *statement_prior_year_list() );

/* Process */
for STATEMENT_PRIOR_YEAR *statement_prior_year in
    statement_prior_year_list
{
    ACCOUNT *prior_account =
       account_element_list_seek(
          account_name,
          statement_prior_year->
             element_statement_list );

    list_set(
       data_list,
       statement_prior_year_account_data(
          prior_account ) );
}

/* Usage */
char *statement_prior_year_account_data(
    ACCOUNT *prior_account );

/* Process */
if ( !prior_account
||  !prior_account->account_journal_latest->balance )
{
    boolean cell_empty = 1;
    int delta_prior_percent = 0;
    double prior_year_amount = 0.0;
}
else
{
    int delta_prior_percent =
       prior_account->delta_prior_percent;

    double prior_year_amount =
       prior_account->
          account_journal_latest->
          balance;
}

char *statement_prior_year_cell_display(
    cell_empty,
    delta_prior_percent,
    prior_year_amount );

/* Attributes */
char *date_time_string;
LIST *element_statement_list;
```

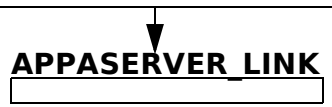


STATEMENT_PRIOR_YEAR (2)

```
/* Usage */
LIST *statement_prior_year_subclassification_data_list(
   char *subclassification_name,
   LIST *statement_prior_year_list() );

/* Process */
for STATEMENT_PRIOR_YEAR *statement_prior_year in
   statement_prior_year_list
{
   SUBCLASSIFICATION *prior_subclassification =
      subclassification_element_list_seek(
         subclassification_name,
         statement_prior_year->
            element_statement_list );

   list_set(
      data_list,
      statement_prior_year_subclassification_data(
         prior_subclassification ) );
}

/* Usage */
char *statement_prior_year_subclassification_data(
   SUBCLASSIFICATION *prior_subclassification );

/* Process */
if (!prior_subclassification
||  !prior_subclassification->sum )
{
   boolean cell_empty = 1;
}
else
{
   int delta_prior_percent =
      prior_subclassification->delta_prior_percent;

   double prior_year_amount =
      prior_subclassification->sum;
}

char *statement_prior_year_cell_display(
   cell_empty,
   delta_prior_percent,
   prior_year_amount );

/* Usage */
LIST *statement_prior_year_element_data_list(
   char *element_name,
   LIST *statement_prior_year_list() );

/* Process */
for STATEMENT_PRIOR_YEAR *statement_prior_year in
   statement_prior_year_list
{
   ELEMENT *prior_element =
      element_seek(
         element_name,
         statement_prior_year->
            element_statement_list );

   list_set(
      data_list,
      statement_prior_year_element_data(
         prior_element ) );
}

/* Usage */
char *statement_prior_year_element_data(
   ELEMENT *prior_element );

/* Process */
if ( !prior_element || !prior_element->sum )
{
   boolean cell_empty = 1;
}
else
{
   int delta_prior_percent =
      prior_element->delta_prior_percent;

   double prior_year_amount =
      prior_element->sum;
}

char *statement_prior_year_cell_display(
   cell_empty,
   delta_prior_percent,
   prior_year_amount );

/* Usage */
char *statement_prior_year_cell_display(
   boolean cell_empty,
   int delta_prior_percent,
   double prior_year_amount );

/* Process */
if ( cell_empty )
{
   *cell_display[] = '\0';
}
else
{
   sprintf(
      cell_display[],
      "%d%c %s",
      delta_prior_percent,
      '%',
      string_commas_dollar(
         prior_year_amount );
}

/* Usage */
LIST *statement_prior_year_heading_list(
   LIST *statement_prior_year_list );

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

for STATEMENT_PRIOR_YEAR *statement_prior_year in
   statement_prior_year_list
{
   char *statement_date_american(
      statement_prior_year->date_time_string );

   list_set(
      heading_list,
      statement_date_american() );
}
```

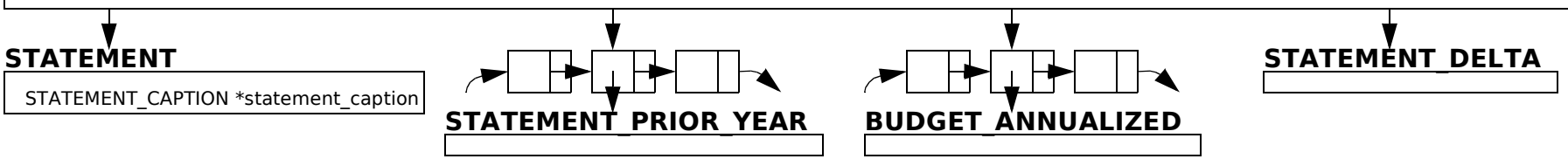
STATEMENT_LINK

```
/* Usage */
STATEMENT_LINK *statement_link_new(
    char *application_name,
    char *process_name,
    char *appaserver_parameter_data_directory,
    char *transaction_begin_date_string(),
    char *transaction_date_time_closing(),
    pid_t process_id );

/* Process */
STATEMENT_LINK *statement_link_calloc(
    void );

APPASERVER_LINK *appaserver_link =
    appaserver_link_new(
      application_http_prefix( application_name ),
      application_server_address(
        application_ssl_support_boolean(
          application_name ) ),
      appaserver_parameter_data_directory,
      process_name /* filename_stem*/,
      application_name,
      process_id,
      (char *)0 /* session_key */,
      transaction_begin_date_string
        /* begin_date_string */,
      transaction_date_time_closing
        /* end_date_string */,
      "tex" /* extension */ );

char *tex_filename =
    char *appaserver_link_filename(
      appaserver_link->filename_stem,
      appaserver_link->begin_date_string,
      appaserver_link->end_date_string,
      appaserver_link->process_id,
      appaserver_link->session_key,
      appaserver_link->extension );

char *statement_link_tex_prompt(
    char *process_name );

char *tex_anchor_html =
    appaserver_link_anchor_html(
      appaserver_link->
        appaserver_link_prompt->
        filename /* prompt_filename */,
      process_name /* target_window */,
      statement_link_tex_prompt() );

appaserver_link->extension = "pdf";

appaserver_link->appaserver_link_prompt =
    APPASERVER_LINK_PROMPT *
      appaserver_link_prompt_new(
        APPASERVER_LINK_PROMPT_DIRECTORY
          /* probably appaserver_data */,
        appaserver_link->http_prefix,
        appaserver_link->server_address,
        application_name,
        appaserver_link->filename_stem,
        appaserver_link->begin_date_string,
        appaserver_link->end_date_string,
        appaserver_link->process_id,
        appaserver_link->session_key,
        appaserver_link->extension );

char *statement_link_pdf_prompt(
    char *process_name );

char *pdf_anchor_html =
    appaserver_link_anchor_html(
      appaserver_link->
        appaserver_link_prompt->
        filename /* prompt_filename */,
      process_name /* target_window */,
      statement_link_pdf_prompt() );

char *appaserver_link_working_directory(
    appaserver_parameter_data_directory,
    application_name );

/* Attributes */
APPASERVER_LINK *appaserver_link;
char *tex_filename;
char *tex_anchor_html;
char *pdf_anchor_html;
char *appaserver_link_working_directory;
```



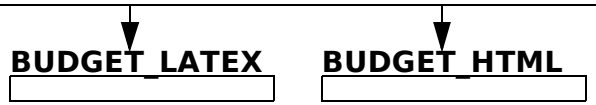
BUDGET (1)

```
/* Usage */
BUDGET *budget_fetch(
    char *application_name,
    char *session_key,
    char *process_name,
    char *output_medium_string,
    char *appaserver_parameter_data_directory );

/* Process */
BUDGET *budget_calloc(
    void );

enum statement_output_medium =
    statement_resolve_output_medium(
        output_medium_string );

DATE *date_now_new(
    date_utc_offset() );

char *date_display_yyyy_mm_dd(
    date_now_new() );

char *transaction_date_begin_date_string(
    TRANSACTION_TABLE,
    date_display_yyyy_mm_dd() );

DATE *budget_begin_date(
    char *transaction_date_begin_date_string() );

int budget_days_so_far(
    DATE *date_now_new(),
    DATE *budget_begin_date() );

int budget_days_in_year(
    DATE *budget_begin_date() );

double budget_year_ratio(
    int budget_days_so_far(),
    int budget_days_in_year() );

LIST *budget_element_name_list(
    char *ELEMENT_REVENUE,
    char *ELEMENT_EXPENSE );

boolean transaction_date_close_boolean(
    TRANSACTION_TABLE,
    TRANSACTION_DATE_CLOSE_TIME,
    TRANSACTION_DATE_MEMO,
    date_display_yyyy_mm_dd() );

char *transaction_date_close_date_time_string(
    TRANSACTION_DATE_PRECLOSE_TIME,
    TRANSACTION_DATE_CLOSE_TIME,
    date_display_yyyy_mm_dd()
        /* transaction_date_as_of */ ),
    transaction_date_close_boolean()
      /* preclose_time_boolean */ );

STATEMENT *statement =
    statement_fetch(
      application_name,
      process_name,
      0 /* prior_year_count */,
      budget_element_name_list(),
      transaction_date_begin_date_string(),
      transaction_date_close_date_time_string()
        /* end_date_time_string */,
      0 /* not fetch_transaction */ );

void element_list_account_statement_list_set(
    statement->element_statement_list );

char *budget_end_date_time_string(
    char *transaction_date_begin_date_string() );

LIST *statement_prior_year_list(
    budget_element_name_list(),
    budget_end_date_time_string(),
    1 /* prior_year_count */,
    statement );

DATE *budget_forecast_date(
    budget_begin_date(),
    budget_days_in_year() );

char *budget_forecast_date_string(
    budget_forecast_date() );

char *budget_forecast_julian_string(
    budget_forecast_date_string() );

char *appaserver_link_working_directory(
    appaserver_parameter_data_directory,
    application_name );

LIST *budget_annualized_list(
    application_name,
    session_key,
    process_name,
    appaserver_parameter_data_directory,
    transaction_date_begin_date_string(),
    statement->element_statement_list,
    list_first( statement_prior_year_list() )
      /* statement_prior_year */,
    budget_forecast_date_string(),
    budget_forecast_julian_string(),
    appaserver_link_working_directory() );

double budget_amount_net(
    budget_annualized_list() );

int budget_annualized_amount_net(
    budget_annualized_list() );

int budget_annualized_budget_net(
    budget_annualized_list() );

STATEMENT_DELTA *statement_delta_new(
      (double)budget_annualized_budget_net()
        /* base_value */,
      (double)budget_annualized_amount_net()
        /* change_value */ );
```



BUDGET (2)

```
/* Process (continued) */
if ( statement_output_medium ==
       statement_output_PDF )
{
   char *budget_year_percent_sub_title(
      statement->
         statement_caption->
         combined,
      budget_year_ratio() );

   BUDGET_LATEX *budget_latex =
      budget_latex_new(
         application_name,
         process_name,
         appaserver_parameter_data_directory,
         statement->
            transaction_date_begin_date_string,
         statement->
            end_date_time_string,
         statement->
            statement_caption->
            logo_filename,
         budget_year_percent_sub_title(),
         budget_annualized_list(),
         budget_amount_net(),
         budget_annualized_amount_net(),
         budget_annualized_budget_net(),
         budget_statement_delta->cell_string,
         getpid() /* process_id */ );
}
else
if ( statement_output_medium ==
       statement_output_table )
{
   char *budget_year_percent_sub_title(
      statement->
         statement_caption->
         sub_title,
      budget_year_ratio() );

   BUDGET_HTML *budget_html =
      budget_html_new(
         budget_year_percent_sub_title(),
         budget_annualized_list(),
         budget_net_amount(),
         budget_annualized_amount_net(),
         budget_annualized_budget_net(),
         budget_statement_delta->cell_string );
}

/* Attributes */
enum statement_output_medium
    statement_output_medium;
DATE *as_of_date;
char *transaction_date_begin_date_string;
DATE *begin_date;
int days_so_far;
int days_in_year;
double year_ratio;
LIST *element_name_list;
boolean transaction_date_close_boolean;
char *transaction_date_close_date_time_string;
STATEMENT *statement;
char *end_date_time_string;
LIST *statement_prior_year_list;
DATE *forecast_date;
char *forecast_date_string;
char *forecast_julian_string;
char *appaserver_link_working_directory;
LIST *budget_annualized_list;
double amount_net;
int annualized_amount_net;
int annualized_budget_net;
STATEMENT_DELTA *statement_delta;
char *year_percent_subtitle;
BUDGET_LATEX *budget_latex;
BUDGET_HTML *budget_html;
```

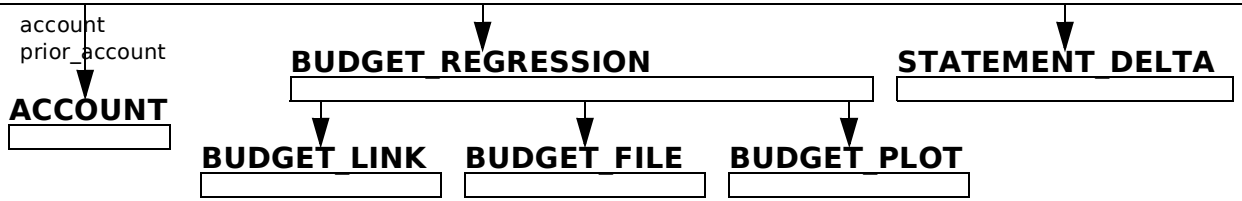


BUDGET (3)

```
/* Usage */
char *budget_display(
    int budget_integer );

/* Process */
if ( budget_integer )
{
    return
        strdup(
            string_commas_integer( budget_integer ) );
}
else
{
    return (char *)0;
}

/* Usage */
char *budget_year_percent_sub_title(
    char *input_subtitle,
    double budget_year_ratio );

/* Process */
int float_ratio_to_percent(
    budget_year_ratio );

snprintf(
    sub_title[],
    sizeof ( sub_title ),
    "%s; Year Percent: %d%c",
    input_subtitle,
    float_ratio_to_percent(),
    '%' );

/* Usage */
DATE *budget_forecast_date(
    DATE *budget_begin_date,
    int budget_days_in_year );

/* Process */
DATE *date_copy( (DATE *)0, budget_begin_date );

DATE *date_increment_days(
    date_copy(),
    budget_days_in_year - 1 );

/* Usage */
char *budget_forecast_date_string(
    DATE *budget_forecast_date );

/* Process */

/* Usage */
char *budget_forecast_julian_string(
    char *budget_forecast_date_string );

/* Process */
snprintf(
    system_string[],
    sizeof ( system_string ),
    "echo %s | date2julian.e 0",
    budget_forecast_date_string );

char *string_fetch( system_string );

/* Usage */
double budget_amount_net(
    LIST *budget_annualized_list );

/* Process */
double revenue_sum =
    budget_amount_sum(
        ELEMENT_REVENUE,
        budget_annualized_list );

double expense_sum =
    budget_amount_sum(
        ELEMENT_EXPENSE,
        budget_annualized_list );

return revenue_sum - expense_sum;

/* Usage */
double budget_amount_sum(
    const char *element_name,
    LIST *budget_annualized_list );

/* Process */

/* Driver */
printf(
    "%s\n",
    budget->
        statement->
        statement_caption->
            frame_title );

if ( budget->budget_latex )
{
    void latex_table_output(
        budget->
            budget_latex->
            statement_link->
            tex_filename,
        budget->
            budget_latex->
            statement_link->
            appaserver_link_working_directory,
        budget->
            budget_latex->
            statement_link->
            pdf_anchor_html,
        budget->
            budget_latex->
            latex,
        budget->
            budget_latex->
            latex_table );
}
else
if ( budget->budget_html )
{
    void statement_html_output(
        budget->budget_html->html_table,
        (char *)0 /* secondary_title */ );
}

for BUDGET_ANNUALIZED *budget_annualized in
    budget->budget_annualized_list
{
    if ( budget_annualized->
            budget_regression->
            confidence_integer )
    {
        printf( "<br>%s\n",
                budget_annualized->
                    budget_regression->
                    budget_link->
                    pdf_anchor_html );
    }
}
```

BUDGET_ANNUALIZED (1)

```
/* Usage */
LIST *budget_annualized_list(
    char *application_name,
    char *session_key,
    char *process_name,
    char *appaserver_parameter_data_directory,
    char *transaction_date_begin_date_string,
    LIST *element_statement_list,
    STATEMENT_PRIOR_YEAR *statement_prior_year,
    char *budget_forecast_date_string(),
    char *budget_forecast_julian_string,
    char *appaserver_link_working_directory );

/* Process */
for ELEMENT *element in element_statement_list
{
    for ACCOUNT *account in
       element->account_statement_list
    {
       BUDGET_ANNUALIZED *
          budget_annualized_new(
             application_name,
             session_key,
             process_name,
             appaserver_parameter_data_directory,
             transaction_begin_date_string,
             element->element_name,
             account,
             statement_prior_year,
             budget_forecast_date_string,
             budget_forecast_julian_string,
             appaserver_link_working_directory );
          list_set_order(
             list,
             budget_annualized_new(),
             budget_annualized_compare_function );
       }
}

/* Usage */
BUDGET_ANNUALIZED *budget_annualized_new(
    char *application_name,
    char *session_key,
    char *process_name,
    char *appaserver_parameter_data_directory,
    char *transaction_begin_date_string,
    char *element_name,
    ACCOUNT *account,
    STATEMENT_PRIOR_YEAR *statement_prior_year,
    char *budget_forecast_date_string,
    char *budget_forecast_julian_string,
    char *appaserver_link_working_directory );

/* Process */
BUDGET_ANNUALIZED *budget_annualized_calloc(
    void );

BUDGET_REGRESSION *budget_regression_fetch(
    application_name,
    session_key,
    process_name,
    appaserver_parameter_data_directory,
    transaction_date_begin_date_string,
    account->account_name,
    budget_forecast_date_string,
    budget_forecast_julian_string,
    appaserver_link_working_directory );

int budget_annualized_budget_integer(
    statement_prior_year,
    account->account_name,
    account->annual_budget );

int budget_annualized_amount_integer(
    account->
       account_journal_latest->
       balance /* account_amount */,
    budget_regression_fetch()->forecast_integer,
    budget_annualized_budget_integer() );

STATEMENT_DELTA *statement_delta_new(
    (double)budget_annualized_budget_integer()
       /* base_value */,
    (double)budget_annualized_amount_integer()
       /* change_value */ );

/* Attributes */
char *element_name;
ACCOUNT *account;
BUDGET_REGRESSION *budget_regression;
int budget_integer;
int amount_integer;
STATEMENT_DELTA *statement_delta;
```

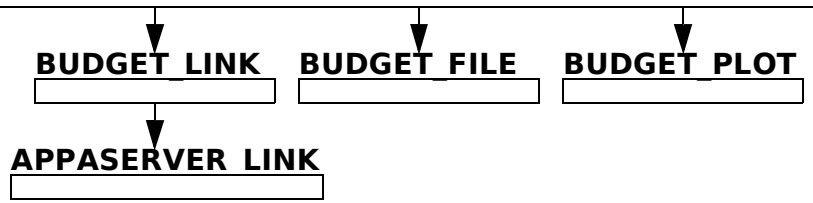


BUDGET_ANNUALIZED (2)

```
/* Usage */
int budget_annualized_amount_net(
   budget_annualized_list() );

/* Process */
int revenue_sum =
   budget_annualized_amount_sum(
      ELEMENT_REVENUE,
      budget_annualized_list );

int expense_sum =
   budget_annualized_amount_sum(
      ELEMENT_EXPENSE,
      budget_annualized_list );

return revenue_sum - expense_sum;

/* Usage */
int budget_annualized_amount_sum(
   const char *element_name,
   LIST *budget_annualized_list );

/* Process */
for BUDGET_ANNUALIZED *budget_annualized in
   budget_annualized_list
{
   if ( string_strcmp(
      budget_annualized->element_name,
      element_name ) == 0 )
   {
      int sum += budget_annualized->amount;
   }
}

/* Usage */
int budget_annualized_budget_net(
   budget_annualized_list() );

/* Process */
int revenue_sum =
   budget_annualized_budget_sum(
      ELEMENT_REVENUE,
      budget_annualized_list );

int expense_sum =
   budget_annualized_budget_sum(
      ELEMENT_EXPENSE,
      budget_annualized_list );

return revenue_sum - expense_sum;

/* Usage */
int budget_annualized_budget_sum(
   const char *element_name,
   LIST *budget_annualized_list );

/* Process */

/* Usage */
int budget_annualized_budget_integer(
   STATEMENT_PRIOR_YEAR *statement_prior_year,
   char *account->account_name,
   int account_annual_budget );

/* Process */
if ( account_annual_budget )
{
   int budget_integer = account_annual_budget;
}
else
if ( statement_prior_year )
{
   ACCOUNT *prior_account =
      account_element_list_seek(
         account_name,
         statement_prior_year->
            element_statement_list );

   if ( prior_account )
   {
      int budget_integer =
         float_round_integer(
            prior_account->
               account_journal_latest->
               balance );
   }
}

/* Usage */
int budget_annualized_amount_integer(
   double account_amount,
   int forecast_integer,
   int budget_annualized_budget_integer );

/* Process */
if ( forecast_integer )
      int amount_integer = forecast_integer;
else
      int amount_integer =
         budget_annualized_budget_integer;

if ( (double)amount_integer < account_amount )
{
   int amount_integer =
      float_round_integer(
         amount_amount );
}

/* Usage */
int budget_annualized_compare_function(
   BUDGET_ANNUALIZED *from_list_budget_annualized,
   BUDGET_ANNUALIZED *compare_budget_annualized );

/* Process */
if (compare_budget_annualized->amount_integer ==
   from_list_budget_annualized->amount_integer )
{
   return 0;
}
else
if (compare_budget_annualized->amount_integer >
   from_list_budget_annualized->amount_integer )
{
   return 1;
}
else
{
   return -1;
}
```

BUDGET_REGRESSION (1)

```
/* Usage */
BUDGET_REGRESSION *budget_regression_fetch(
    char *application_name,
    char *session_key,
    char *process_name,
    char *appaserver_parameter_data_directory,
    char *transaction_date_begin_date_string,
    char *account_name,
    char *budget_forecast_date_string,
    char *budget_forecast_julian_string,
    char *appaserver_link_working_directory );

/* Process */
BUDGET_REGRESSION *budget_regression_calloc(
    void );

BUDGET_LINK *budget_link_new(
    application_name,
    session_key,
    process_name,
    appaserver_parameter_data_directory,
    account_name );

BUDGET_FILE *budget_file_new(
    session_key,
    appaserver_link_working_directory );

char *budget_regression_sql(
    JOURNAL_TABLE,
    transaction_date_begin_date_string,
    account_name );

char *budget_regression_text_system_string(
    char *budget_file_new()->text_specification,
    char *budget_regression_sql() );

system( budget_regression_text_system_string() );

int budget_regression_row_count(
    char *budget_file_new()->text_specification );

if ( budget_regression_row_count() < 3 ) return this;

char *budget_regression_julian_system_string(
    BUDGET_REGRESSION_DATE_LABEL,
    BUDGET_REGRESSION_BALANCE_LABEL,
    budget_file_new()->text_specification,
    budget_file_new()->julian_specification );

system( budget_regression_julian_system_string() );

char *budget_regression_forecast_system_string(
    BUDGET_REGRESSION_DATE_LABEL,
    BUDGET_REGRESSION_BALANCE_LABEL,
    budget_forecast_julian_string,
    budget_file_new()->julian_specification,
    budget_file_new()->regression_specification );

char *string_fetch(
    budget_regression_forecast_system_string() );

int budget_regression_forecast_integer(
    char *string_fetch() );

if ( budget_regression_forecast_integer() )
{
    int budget_regression_confidence_integer(
        char *string_fetch() );

    BUDGET_PLOT *budget_plot_new(
        BUDGET_REGRESSION_DATE_LABEL,
        BUDGET_REGRESSION_BALANCE_LABEL,
        session_key,
        account_name,
        budget_forecast_date_string,
        budget_forecast_julian_string,
        appaserver_link_working_directory,
        budget_link->pdf_filename,
        budget_file->julian_specification /* will append */,
        budget_regression_forecast_integer(),
        budget_regression_confidence_integer() );
}

system(
    budget_file_new()->
        rm_text_system_string );

system(
    budget_file_new()->
        rm_julian_system_string );

system(
    budget_file_new()->
        rm_regression_system_string );

/* Attributes */
char *account_name;
BUDGET_LINK *budget_link;
BUDGET_FILE *budget_file;
char *sql;
char *text_system_string;
int row_count;
char *julian_system_string;
char *forecast_system_string;
char *string_fetch;
int forecast_integer;
int confidence_integer;
BUDGET_PLOT *budget_plot;
```



BUDGET_REGRESSION (2)

```
/* Usage */
char *budget_regression_sql(
    const char *JOURNAL_TABLE,
    char *transaction_date_begin_date_string,
    char *account_name );

/* Process */
char *budget_regression_where(
    char *transaction_date_begin_date_string,
    char *account_name );

snprintf(
    sql[],
    sizeof ( sql ),
    "select "
    "substr( transaction_date_time, 1, 10 ), "
    "balance "
    "from %s "
    "where %s "
    "order by transaction_date_time",
    JOURNAL_TABLE,
    budget_regression_where() );

/* Usage */
char *budget_regression_julian_system_string(
    const char *BUDGET_REGRESSION_DATE_LABEL,
    const char *BUDGET_REGRESSION_BALANCE_LABEL,
    char *text_specification,
    char *julian_specification );

/* Process */
snprintf(
    julian_system_string[],
    sizeof ( julian_system_string ),
    "(\n"
    "echo '%s,%s'\n"
    "cat %s |\n"
    "date2julian.e 0 ','\n"
    ") | cat > %s",
    BUDGET_REGRESSION_DATE_LABEL,
    BUDGET_REGRESSION_BALANCE_LABEL,
    text_specification,
    julian_specification );

/* Usage */
char *budget_regression_forecast_system_string(
    const char *BUDGET_REGRESSION_DATE_LABEL,
    const char *BUDGET_REGRESSION_BALANCE_LABEL,
    char *budget_forecast_julian_string,
    char *julian_specification,
    char *regression_specification );

/* Process */
snprintf(
    forecast_system_string[],
    sizeof ( forecast_system_string ),
    "linear_regression_forecast.sh %s %s %s %s %s",
    julian_specification,
    regression_specification,
    BUDGET_REGRESSION_DATE_LABEL,
    BUDGET_REGRESSION_BALANCE_LABEL,
    budget_forecast_julian_string );
```

BUDGET_PLOT

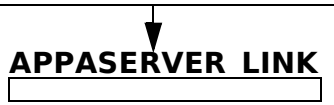
```
/* Usage */
BUDGET_PLOT *budget_plot_new(
    const char *BUDGET_REGRESSION_DATE_LABEL,
    const char *BUDGET_REGRESSION_BALANCE_LABEL,
    char *session_key,
    char *account_name,
    char *budget_forecast_date_string,
    char *budget_forecast_julian_string,
    char *appaserver_link_working_directory,
    char *budget_link->pdf_filename,
    char *budget_file->julian_specification
        /* will append */,
    int forecast_integer,
    int confidence_integer );

/* Process */
BUDGET_PLOT *budget_file_calloc(
    void );

FILE *appaserver_append_file(
    julian_specification );

fprintf(
    appaserver_append_file(),
    "%s,%d\n",
    budget_forecast_julian_string,
    forecast_integer );

fclose( appaserver_append_file() );

chdir( appaserver_link_working_directory );

void budget_plot_write_Rscript(
    BUDGET_REGRESSION_DATE_LABEL,
    BUDGET_REGRESSION_BALANCE_LABEL,
    BUDGET_PLOT_RSCRIPT_FILENAME,
    account_name,
    budget_forecast_date_string,
    julian_specification,
    confidence_integer );

char *budget_plot_system_string(
    BUDGET_PLOT_RSCRIPT_FILENAME );

system( budget_plot_system_string() );

char *budget_plot_rename_system_string(
    BUDGET_PLOT_DEFAULT_PDF,
    pdf_filename );

system( budget_plot_rename_system_string() );

char *budget_plot_remove_system_string(
    const char *BUDGET_PLOT_RSCRIPT_FILENAME );

system( budget_plot_remove_system_string() );

/* Usage */
void budget_plot_write_Rscript(
    const char *BUDGET_REGRESSION_DATE_LABEL,
    const char *BUDGET_REGRESSION_BALANCE_LABEL,
    const char *BUDGET_PLOT_RSCRIPT_FILENAME,
    char *account_name,
    char *budget_forecast_date_string,
    char *julian_specification,
    int confidence_integer );

/* Process */
FILE *appaserver_output_file(
    BUDGET_PLOT_RSCRIPT_FILENAME );

fprintf(
    appaserver_output_file(),
    "dataframe = read.csv( '%s' )\n"
    "x = dataframe[[ '%s' ]]\n"
    "y = dataframe[[ '%s' ]]\n"
    "linearModel <- lm( y ~ x )\n"
    "message <-\n"
    "\tpaste(\n"
    "\t\t'Forecast %s = $',\n"
    "\t\tmax( y ),\n"
    "\t\t', Confidence = '\n"
    "\t\t'%d%c')\n"
    "plot(\n"
    "\tx,\n"
    "\ty,\n"
    "\tmain = '%s forecast',\n"
    "\tsub = message,\n"
    "\txlab = 'Transaction Date (Julian)',\n"
    "\tylab = '%s',\n"
    "\tabline( linearModel ) )\n",
    julian_specification,
    BUDGET_REGRESSION_DATE_LABEL,
    BUDGET_REGRESSION_BALANCE_LABEL,
    budget_forecast_date_string,
    confidence_integer,
    '%',
    account_name,
    account_name );

fclose( appaserver_output_file() );

/* Usage */
char *budget_plot_system_string(
    const char *BUDGET_PLOT_RSCRIPT_FILENAME );

/* Process */
snprintf(
    system_string[],
    sizeof ( system_string ),
    "Rscript.sh %s",
    BUDGET_PLOT_RSCRIPT_FILENAME );

/* Usage */
char *budget_plot_rename_system_string(
    const char *BUDGET_PLOT_DEFAULT_PDF,
    char *pdf_filename );

/* Process */
snprintf(
    system_string[],
    sizeof ( system_string ),
    "mv %s %s",
    BUDGET_PLOT_DEFAULT_PDF,
    pdf_filename );

/* Attributes */
```

BUDGET_FILE

```
/* Usage */
BUDGET_FILE *budget_file_new(
    char *session_key,
    char *appaserver_link_working_directory );

/* Process */
BUDGET_FILE *budget_file_calloc(
    void );

char *text_specification =
    budget_file_specification(
        "txt" /* extension */,
        session_key,
        appaserver_link_working_directory );

char *rm_text_system_string =
    budget_file_rm_system_string(
        text_specification );

char *julian_specification =
    budget_file_specification(
        "csv" /* extension */,
        session_key,
        appaserver_link_working_directory );

char *rm_julian_system_string =
    budget_file_rm_system_string(
        julian_specification );

char *regression_specification =
    budget_file_specification(
        "dat" /* extension */,
        session_key,
        appaserver_link_working_directory );

char *rm_regression_system_string =
    budget_file_rm_system_string(
        regression_specification );

/* Usage */
char *budget_file_specification(
    const char *extension,
    char *session_key,
    char *appaserver_link_working_directory );

/* Process */

/* Usage */
char *budget_file_rm_system_string(
    char *budget_file_specification() );

/* Process */

/* Attributes */
char *text_specification;
char *rm_text_system_string;
char *julian_specification;
char *rm_julian_system_string;
char *regression_specification;
char *rm_regression_system_string;
```

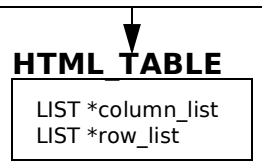
BUDGET_LINK

```
/* Usage */
BUDGET_LINK *budget_link_new(
    char *application_name,
    char *session_key,
    char *process_name,
    char *appaserver_parameter_data_directory,
    char *account_name );

/* Process */
BUDGET_LINK *budget_link_calloc(
    void );

char *budget_link_begin_date_string();

APPASERVER_LINK *appaserver_link =
    APPASERVER_LINK *appaserver_link_new(
        application_http_prefix(
            application_ssl_support_boolean(
                application_name ) ),
        application_server_address(),
        appaserver_parameter_data_directory,
        process_name /* filename_stem */,
        application_name,
        (pid_t)0 /* process_id */,
        session_key,
        budget_link_begin_date_string(),
        (char *)0 /* end_date_string */,
        “pdf” /* extension */ );

char *pdf_filename =
    char *appaserver_link_filename(
        appaserver_link->filename_stem,
        appaserver_link->begin_date_string,
        appaserver_link->end_date_string,
        appaserver_link->process_id,
        appaserver_link->session_key,
        appaserver_link->extension );

APPASERVER_LINK_PROMPT *
    appaserver_link->appaserver_link_prompt =
        appaserver_link_prompt_new(
            APPASERVER_LINK_PROMPT_DIRECTORY
                /* probably appaserver_data */,
            appaserver_link->http_prefix,
            appaserver_link->server_address,
            application_name,
            appaserver_link->filename_stem,
            appaserver_link->begin_date_string,
            appaserver_link->end_date_string,
            appaserver_link->process_id,
            appaserver_link->session_key,
            appaserver_link->extension );

char *budget_link_prompt_string(
    const char *BUDGET_LINK_PROMPT,
    char *account_name );

char *pdf_anchor_html =
    char *appaserver_link_anchor_html(
        appaserver_link->
            appaserver_link_prompt->
            filename /* prompt_filename */,
        process_name /* target_window */,
        budget_link_prompt_string() );

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

/* Process */
static int count;

snprintf(
    static begin_date_string[],
    sizeof ( begin_date_string ),
    “%d”,
    ++count );

/* Attributes */
APPASERVER_LINK *appaserver_link;
char *pdf_filename;
char *pdf_anchor_html;
```



BUDGET_HTML (1)

```
/* Usage */
BUDGET_HTML *budget_html_new(
    char *budget_year_percent_sub_title(),
    LIST *budget_annualized_list(),
    double budget_amount_net(),
    int  budget_annualized_amount_net(),
    int budget_annualized_budget_net(),
    char *delta_cell_string );

/* Process */
BUDGET_HTML *budget_html_calloc( void );

HTML_TABLE *budget_html_table(
    budget_year_percent_sub_title,
    budget_annualized_list,
    budget_amount_net,
    budget_annualized_amount_net,
    budget_annualized_budget_net,
    delta_cell_string );

/* Usage */
HTML_TABLE *budget_html_table(
    char *budget_year_percent_sub_title,
    LIST *budget_annualized_list,
    double budget_amount_net,
    int budget_annualized_amount_net,
    int budget_annualized_budget_net,
    char *delta_cell_string );

/* Process */
HTML_TABLE *html_table =
    html_table_new(
        (char *)0 /* title */,
        budget_year_percent_sub_title,
        (char *)0 /* sub_sub_title */ );

html_table->column_list =
    budget_html_column_list();

html_table->row_list =
    budget_html_row_list(
        budget_annualized_list );

list_set(
    html_table->row_list,
    HTML_ROW *budget_html_sum_row(
        double budget_amount_net,
        int budget_annualized_amount_net,
        int budget_annualized_budget_net,
        char *delta_cell_string ) );

/* Usage */
LIST *budget_html_column_list(
    void );

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

list_set(
    column_list,
    HTML_COLUMN *html_column_new(
        STATEMENT_ELEMENT_HEADING,
        0 /* not right_Justified_boolean */ ) );

list_set(
    column_list,
    HTML_COLUMN *html_column_new(
        STATEMENT_ACCOUNT_HEADING,
        0 /* not right_Justified_boolean */ ) );

list_set(
    column_list,
        HTML_COLUMN *html_column_new(
            STATEMENT_BALANCE_HEADING,
            1 /* right_Justified_boolean */ ) );

list_set(
    column_list,
    HTML_COLUMN *html_column_new(
        STATEMENT_ROW_COUNT_HEADING,
        1 /* right_justified_boolean */ ) );

list_set(
    column_list,
    HTML_COLUMN *html_column_new(
        STATEMENT_CONFIDENCE_HEADING,
        1 /* right_justified_boolean */ ) );

list_set(
    column_list,
    HTML_COLUMN *html_column_new(
        STATEMENT_ANNUALIZED_HEADING,
        1 /* right_Justified_boolean */ ) );

list_set(
    column_list,
    HTML_COLUMN *html_column_new(
        STATEMENT_BUDGET_HEADING,
        1 /* right_Justified_boolean */ ) );

list_set(
    column_list,
    HTML_COLUMN *html_column_new(
        STATEMENT_DIFFERENCE_HEADING,
        1 /* right_Justified_boolean */ ) );

/* Attribute */
HTML_TABLE *html_table;
```

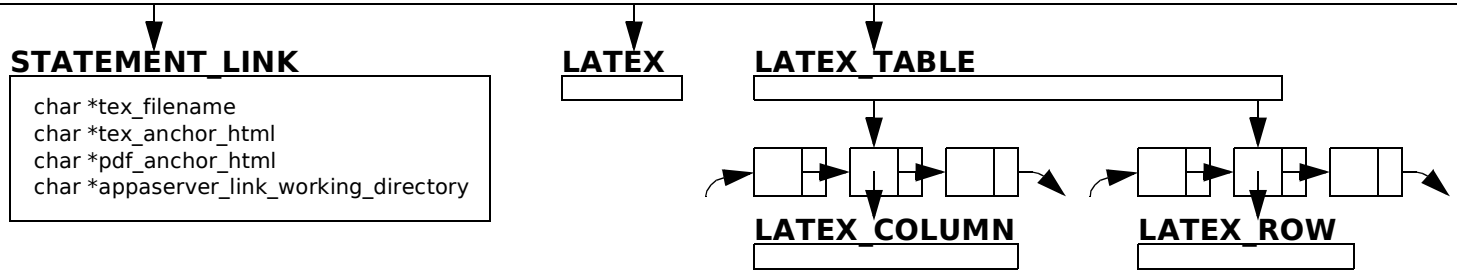


BUDGET_HTML (2)

```
/* Usage */
LIST *budget_html_row_list(
    LIST *budget_annualized_list );

/* Process */
for BUDGET_ANNUALIZED *budget_annualized in
    budget_annualized_list
{
    list_set(
        row_list,
        HTML_ROW *budget_html_row(
            budget_annualized->element_name,
            budget_annualized->
                account->
                account_name,
            budget_annualized->
                account->
                account_journal_latest->
                balance /* account_amount */,
            budget_annualized->
                budget_regression->
                row_count,
            budget_annualized->
                budget_regression->
                confidence,
            budget_annualized->amount_integer,
            budget_annualized->budget_integer,
            budget_annualized->
                statement_delta ) );
}

/* Usage */
HTML_ROW *budget_html_row(
    char *element_name,
    char *account_name,
    double account_amount,
    int row_count,
    int confidence_integer,
    int annualized_amount_integer,
    int annualized_budget_integer,
    STATEMENT_DELTA *statement_delta );

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

char *statement_cell_label_datum(
    element_name /* name */ );

list_set(
    cell_list,
    html_cell_new(
        statement_cell_label_datum(),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

char *statement_cell_label_datum(
    account_name /* name */ );

list_set(
    cell_list,
    html_cell_new(
        statement_cell_label_datum(),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        strdup(
            string_commas_money(
                account_amount ) ),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        strdup(
            string_commas_integer(
            row_count ) ),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        strdup(
            string_commas_integer(
            confidence_integer ) ),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        strdup(
            string_commas_integer(
                annualized_amount_integer ) ),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        budget_display( annualized_budget_integer ),
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

list_set(
    cell_list,
    html_cell_new(
        (statement_delta)
            ? statement_delta->cell_string
            : (char *)0,
        0 /* not large_boolean */,
        0 /* not bold_boolean */ ) );

HTML_ROW *html_row_new( cell_list );
```

BUDGET_LATEX (1)

```
/* Usage */
BUDGET_LATEX *budget_latex_new(
    char *application_name,
    char *process_name,
    char *appaserver_parameter_data_directory,
    char *statement->
       transaction_date_begin_date_string,
    char *statement_end_date_time_string,
    char *statement_caption_logo_filename,
    char *budget_year_percent_sub_title(),
    LIST *budget_annualized_list(),
    double budget_amount_net(),
    int budget_annualized_amount_net(),
    int budget_annualized_budget_net(),
    char *delta_cell_string,
    pid_t process_id );

/* Process */
BUDGET_LATEX *budget_latex_calloc(
    void );

STATEMENT_LINK *statement_link =
    statement_link_new(
       application_name,
       process_name,
       appaserver_parameter_data_directory,
       transaction_date_begin_date_string,
       statement_end_date_time_string,
       process_id );

LATEX *latex =
    latex_new(
       statement_link->tex_filename,
       statement_link->
          appaserver_link_working_directory,
       0 /* not landscape_boolean */,
       statement_caption_logo_filename );

LATEX_TABLE *latex_table =
    budget_latex_table(
       budget_year_percent_sub_title,
       budget_annualized_list,
       budget_amount_net,
       budget_annualized_amount_net,
       budget_annualized_budget_net,
       delta_cell_string );

/* Attributes */
STATEMENT_LINK *statement_link;
LATEX *latex;
LATEX_TABLE *latex_table;;
```



BUDGET_LATEX (2)

```
/* Usage */
LATEX_TABLE *budget_latex_table(
    char *budget_year_percent_sub_title,
    LIST *budget_annualized_list,
    double budget_amount_net(),
    int budget_annualized_amount_net(),
    int budget_annualized_budget_net(),
    char *delta_cell_string );

/* Process */
LIST *budget_latex_colum_list();

LIST *budget_latex_row_list(
    budget_annualized_list,
    budget_latex_column_list() );

list_set(
    budget_latex_row_list(),
    LATEX_ROW *budget_latex_sum_row(
       double budget_amount_net,
       int budget_annualized_amount_net,
       int budget_annualized_budget_net,
       char *delta_cell_string,
       LIST *budget_latex_column_list() ) );

LATEX_TABLE *latex_table_new(
    budget_year_percent_sub_title /* title */,
    budget_latex_column_list(),
    budget_latex_row_list() );

/* Usage */
LIST *budget_latex_column_list( void );

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

list_set(
    latex_column_list,
    latex_column_new(
       STATEMENT_ELEMENT_HEADING
          /* heading_string */,
       latex_column_text /* latex_column_enum */,
       0 /* float_decimal_count */,
       (char *)0 /* paragraph_size */,
       1 /* first_column_list */ ) );

list_set(
    latex_column_list,
    latex_column_new(
       STATEMENT_ACCOUNT_HEADING
          /* heading_string */,
       latex_column_text /* latex_column_enum */,
       0 /* float_decimal_count */,
       (char *)0 /* paragraph_size */,
       0 /* not first_column_boolean */ ) );

list_set(
    latex_column_list,
    latex_column_new(
       STATEMENT_BALANCE_HEADING
          /* heading_string */,
       latex_column_float /* latex_column_enum */,
       0 /* float_decimal_count */,
       (char *)0 /* paragraph_size */,
       0 /* not first_column_boolean */ ) );

list_set(
    latex_column_list,
    latex_column_new(
       STATEMENT_ANNUALIZED_HEADING
          /* heading_string */,
       latex_column_float /* latex_column_enum */,
       0 /* float_decimal_count */,
       (char *)0 /* paragraph_size */,
       0 /* not first_column_boolean */ ) );

list_set(
    latex_column_list,
    latex_column_new(
       STATEMENT_BUDGET_HEADING
          /* heading_string */,
       latex_column_float /* latex_column_enum */,
       0 /* float_decimal_count */,
       (char *)0 /* paragraph_size */,
       0 /* not first_column_boolean */ ) );

list_set(
    latex_column_list,
    latex_column_new(
       STATEMENT_DIFFERENCE_HEADING
          /* heading_string */,
       latex_column_float /* latex_column_enum */,
       0 /* float_decimal_count */,
       (char *)0 /* paragraph_size */,
       0 /* not first_column_boolean */ ) );
```

BUDGET_LATEX (3)

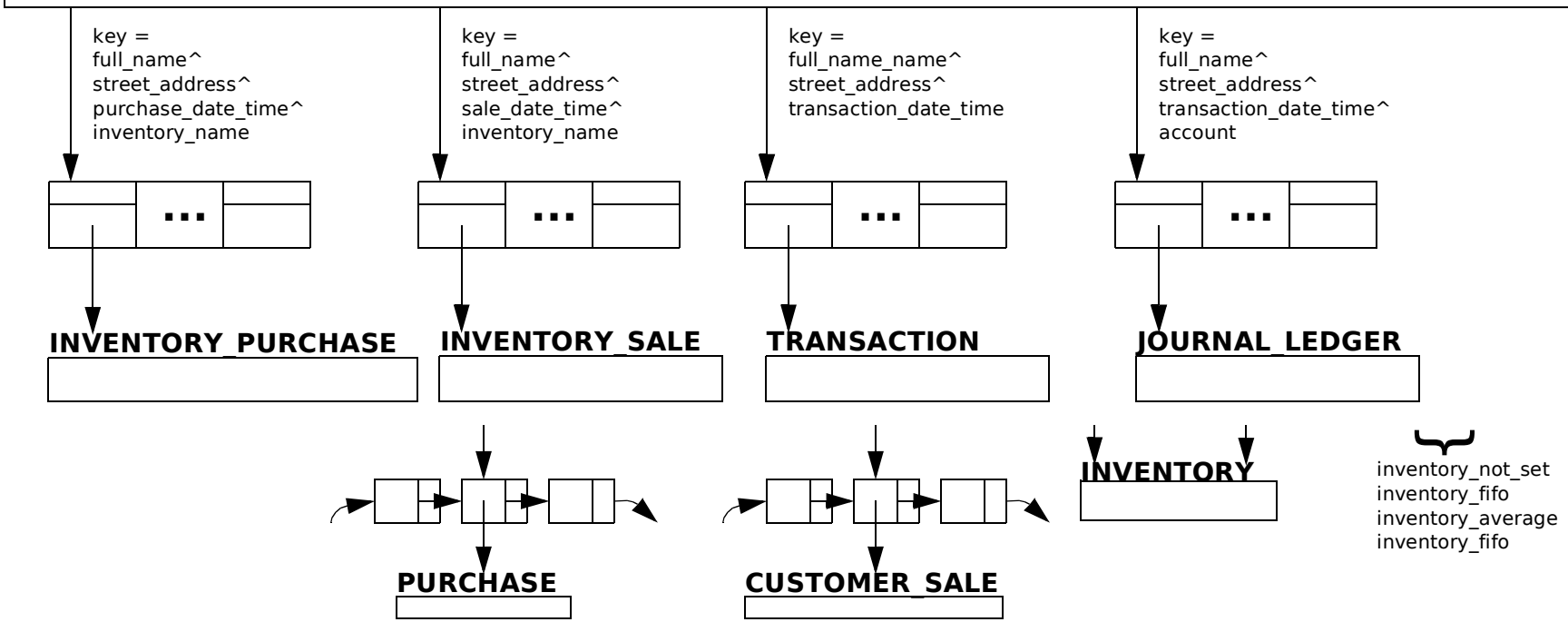
```
/* Usage */
LIST *budget_latex_row_list(
    LIST *budget_annualized_list(),
    LIST *budget_latex_column_list() );

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

for BUDGET_ANNUALIZED *budget_annualized in
    budget_annualized_list
{
    list_set(
        latex_row_list,
        LATEX_ROW *budget_latex_row(
            budget_annualized->element_name,
            budget_annualized->account->account_name,
            budget_annualized->
                account->
                account_journal_latest->
                balance
                    /* account_amount */,
            budget_annualized->amount_integer,
            budget_annualized->budget_integer,
            budget_annualized->statement_delta,
            budget_latex_column_list ) );
}

/* Usage */
LATEX_ROW *budget_latex_row(
    char *element_name,
    char *account_name,
    double account_amount,
    int annualized_amount_integer,
    int annualized_budget_integer,
    STATEMENT_DELTA *statement_delta,
    LIST *latex_column_list );

/* Process */
list_rewind( latex_column_list );

LIST *cell_list = list_new();

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
list_next( latex_column_list );

char *statement_cell_label_datum(
    element_name /* name */ );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        statement_cell_label_datum() ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
list_next( latex_column_list );

char *statement_cell_label_datum(
    account_name /* name */ );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        statement_cell_label_datum() ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
list_next( latex_column_list );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        strdup(
            string_commas_money(
                account_amount ) ) ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
list_next( latex_column_list );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        strdup(
            string_commas_integer(
                annualized_amount_integer ) ) ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
list_next( latex_column_list );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        budget_display(
            annualized_budget_integer ) ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
list_next( latex_column_list );

list_set(
    cell_list,
    latex_cell_small_new(
        latex_column,
        0 /* not first_row_boolean */,
        (statement_delta)
            ? statement_delta->cell_string
            : (char *)0 ) );

LATEX_ROW *latex_row_new( cell_list );
```

ENTITY_SELF_INVENTORY (1)

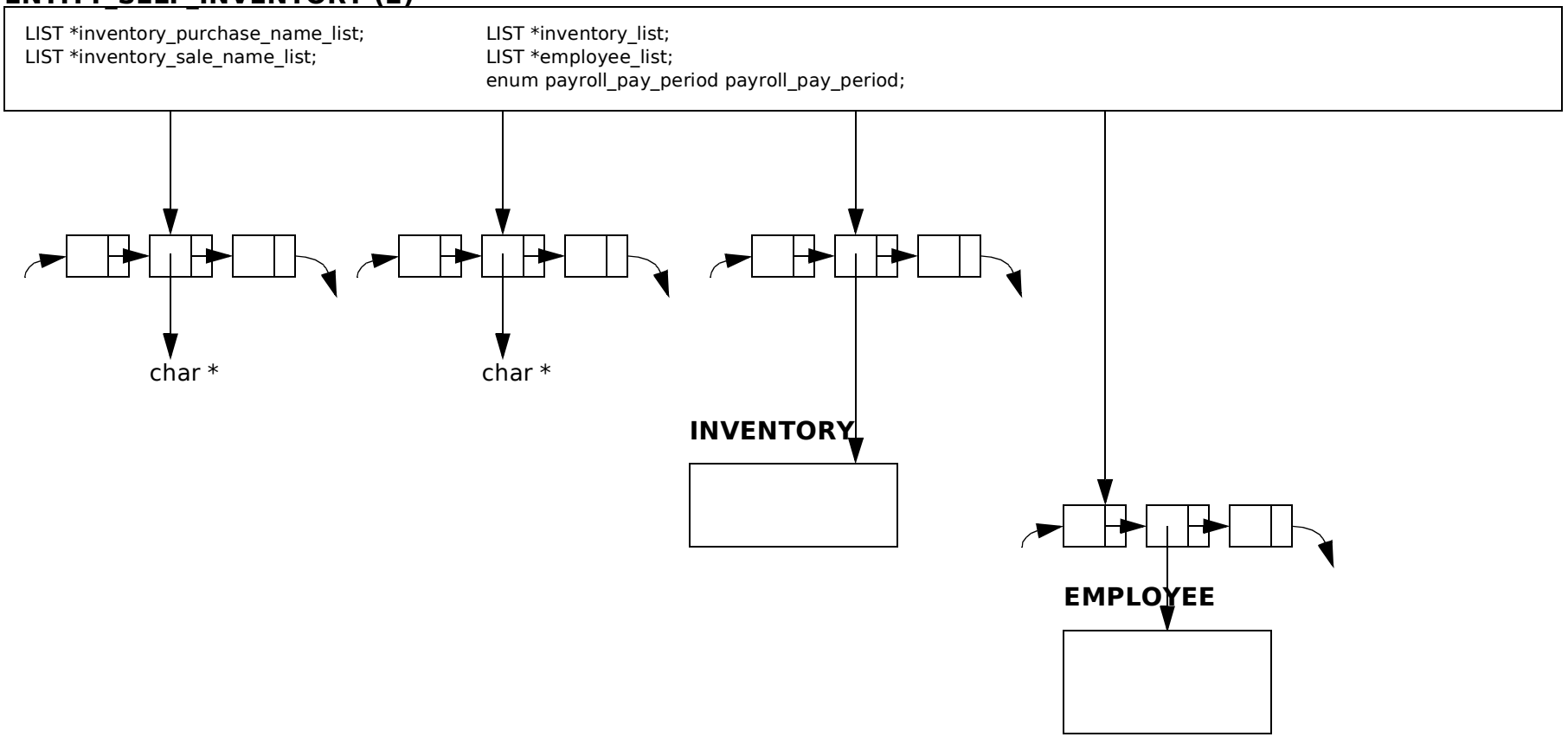
HASH_TABLE *inventory_purchase_hash_table;
HASH_TABLE *inventory_sale_hash_table;
HASH_TABLE *transaction_hash_table;
HASH_TABLE *journal_ledger_hash_table;

LIST *inventory_purchase_list;
LIST *inventory_customer_sale_list;
INVENTORY *purchase_inventory;
INVENTORY *sale_inventory;

enum inventory_cost_method inventory_cost_method;



ENTITY_SELF_INVENTORY (2)



SELF_TAX (1)

```
/* Usage */
SELF_TAX *self_tax_fetch(
    const char *SELF_TAX_TABLE,
    char *full_name,
    char *street_address );

/* Process */
char *entity_primary_where(
    full_name,
    street_address );

char *appaserver_system_string(
    SELF_TAX_SELECT,
    SELF_TAX_TABLE,
    entity_primary_where() );

char *string_pipe_input(
    appaserver_system_string() );

SELF_TAX *self_tax_parse(
    full_name,
    street_address,
    string_pipe_input() );

enum self_tax_resolve_payroll_pay_period(
    char *self_tax_parse()->payroll_pay_period_string );

/* Usage */
SELF_TAX *self_tax_parse(
    char *full_name,
    char *street_address,
    char *string_pipe_input() );

/* Process */
SELF_TAX *self_tax_new(
    full_name,
    street_address );

/* Usage */
SELF_TAX *self_tax_new(
    char *full_name,
    char *street_address );

/* Process */
SELF_TAX *self_tax_calloc(
    void );

/* Attributes */
char *full_name;
char *street_address;
char *inventory_cost_method;
char *payroll_pay_period_string;
char *payroll_begin_day;
double social_security_combined_tax_rate;
int social_security_payroll_ceiling;
double medicare_combined_tax_rate;
double medicare_additional_withholding_rate;
int medicare_additional_gross_pay_floor;
double federal_withholding_allowance_period_value;
double federal_nonresident_withholding_income_premium;
double state_withholding_allowance_period_value;
double state_itemized_allowance_period_value;
int federal_unemployment_wage_base;
double federal_unemployment_tax_standard_rate;
double federal_unemployment_threshold_rate;
double federal_unemployment_tax_minimum_rate;
int state_unemployment_wage_base;
double state_unemployment_tax_rate;
double state_sales_tax_rate;
double energy_charge_kilowatts_per_hour;
char *paypall_cash_account_name;
enum self_tax_payroll_pay_period payroll_pay_period;
```

pay_period_unknown
pay_period_weekly
pay_period_biweekly
pay_period_semimonthly
pay_period_monthly

SELF_TAX (2)

```
/* Usage */
char *self_tax_paypal_cash_account_name(
    char *full_name,
    char *street_address );

/* Process */
SELF_TAX *self_tax_fetch(
    SELF_TAX_TABLE,
    full_name,
    street_address );

self_tax_fetch()->paypall_cash_account_name;

/* Usage */
double self_tax_state_sales_tax_rate(
    char *full_name,
    char *street_address );

/* Process */
SELF_TAX *self_tax_fetch(
    SELF_TAX_TABLE,
    full_name,
    street_address );

self_tax_fetch()->state_sales_tax_rate;

/* Usage */
char *self_tax_resolve_payroll_pay_period_string(
    enum self_tax_payroll_pay_period
      self_tax_payroll_pay_period );

/* Process */
```

SALE (1)

```
/* Usage */
SALE *sale_trigger_new(
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *state,
    char *preupdate_full_name,
    char *preupdate_street_address,
    char *preupdate_uncollectible_date_time );

/* Process */
SALE *sale_calloc(
    void );

SALE_FETCH *sale_fetch_new(
    SALE_SELECT,
    SALE_TABLE,
    full_name,
    street_address,
    sale_date_time );

/* May be deleted */
if ( !sale_fetch_new() ) return null;

if ( list_length( sale_fetch_new()->inventory_sale_list ) )
{
    double inventory_sale_total(
        sale_fetch_new()->inventory_sale_list );

    double inventory_sale_CGS_total(
        sale_fetch_new()->inventory_sale_list );
}

if ( list_length( sale_fetch_new()->
          specific_inventory_sale_list ) )
{
    double specific_inventory_sale_total(
        sale_fetch_new()->specific_inventory_sale_list );

    double specific_inventory_sale_CGS_total(
        sale_fetch_new()->specific_inventory_sale_list );
}

if ( list_length( sale->fixed_service_sale_list ) )
{
    double fixed_service_sale_total(
        sale_fetch_new()->fixed_service_sale_list );
}

if ( list_length( sale->hourly_service_sale_list ) )
{
    double hourly_service_sale_total(
        sale_fetch_new()->hourly_service_sale_list );
}

double SALE_GROSS_REVENUE(
    inventory_sale_total(),
    specific_inventory_sale_total(),
    fixed_service_sale_total(),
    hourly_service_sale_total() );

double SALE_SALES_TAX(
    inventory_sale_total(),
    specific_inventory_sale_total(),
    sale_fetch_new()->self_tax_state_sales_tax_rate );

double SALE_INVOICE_AMOUNT(
    SALE_GROSS_REVENUE(),
    SALE_SALES_TAX(),
    sale_fetch_new()->shipping_charge );

double customer_payment_total(
    sale_fetch_new()->
        customer_payment_list );

double SALE_AMOUNT_DUE(
    SALE_INVOICE_AMOUNT(),
    customer_payment_total() );

SALE_TRANSACTION *sale_transaction_new(
    full_name,
    street_address,
    state,
    preupdate_full_name,
    preupdate_street_address,
    sale_fetch_new()->predictive_title_passage_rule,
    sale_fetch_new()->completed_date_time,
    sale_fetch_new()->shipped_date_time,
    sale_fetch_new()->arrived_date,
    sale_fetch_new()->transaction_date_time
        /* prior_transaction_date_time */,
    sale_fetch_new()->shipping_charge,
    inventory_sale_total(),
    inventory_sale_CGS_total(),
    specific_inventory_sale_total(),
    specific_inventory_sale_CGS_total(),
    SALE_GROSS_REVENUE(),
    SALE_SALES_TAX(),
    SALE_INVOICE_AMOUNT() );

SALE_LOSS_TRANSACTION *
    sale_loss_transaction_new(
        full_name,
        street_address,
        uncollectible_date_time,
        state,
        preupdate_full_name,
        preupdate_street_address,
        preupdate_uncollectible_date_time,
        SALE_AMOUNT_DUE() );

/* Attributes */
char *full_name;
char *street_address;
char *sale_date_time;
SALE_FETCH *sale_fetch;
double inventory_sale_total;
double inventory_sale_CGS_total;
double specific_inventory_sale_total;
double specific_inventory_sale_CGS_total;
double fixed_service_sale_total;
double hourly_service_sale_total;
double gross_revenue;
double sales_tax;
double invoice_amount;
double customer_payment_total;
double amount_due;
SALE_TRANSACTION *sale_transaction;
SALE_LOSS_TRANSACTION *sale_loss_transaction;
```

SALE (2)

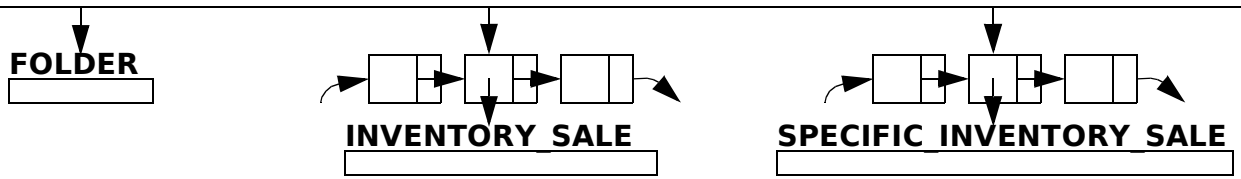
```
/* Usage */
static char *sale_primary_where(
    char *full_name,
    char *street_address,
    char *sale_date_time );

/* Process */

/* Usage */
double SALE_EXTENDED_PRICE(
    double retail_price,
    int quantity,
    double discount_amount );

/* Process */
( A * (double)B ) - C;

/* Usage */
double SALE_GROSS_REVENUE(
    double inventory_sale_total(),
    double specific_inventory_sale_total(),
    double fixed_service_sale_total(),
    double hourly_service_sale_total() );

/* Process */
A + B + C + D;

/* Usage */
double SALE_SALES_TAX(
    double inventory_sale_total(),
    double specific_inventory_sale_total(),
    double self_tax_state_sales_tax_rate() );

/* Process */
(A + B) * C;

/* Usage */
double SALE_INVOICE_AMOUNT(
    double SALE_GROSS_REVENUE(),
    double SALE_SALES_TAX(),
    double shipping_charge );

/* Process */
A + B + C;

/* Usage */
double SALE_AMOUNT_DUE(
    SALE_INVOICE_AMOUNT(),
    customer_payment_total() )

/* Process */
A - B;

/* Usage */
double sale_work_hours(
    char *begin_work_date_time,
    char *end_work_date_time );

/* Process */
DATE *early_date =
    date_19new(
        begin_work_date_time );

DATE *later_date =
    date_19new(
        end_work_date_time );

int date_subtract_minutes(
    later_date,
    earlier_date );

(double)date_subtract_minutes() / 60.0;

/* Usage */
void sale_update(
    const char *SALE_TABLE,
    char *full_name,
    char *street_address,
    char *sale_date_time,
    boolean inventory_sale_boolean,
    boolean specific_inventory_sale_boolean,
    boolean fixed_service_sale_boolean,
    boolean hourly_service_sale_boolean,
    double inventory_sale_total,
    double specific_inventory_sale_total,
    double fixed_service_sale_total,
    double hourly_service_sale_total,
    double sale_gross_revenue,
    double sale_sales_tax,
    double sale_invoice_amount,
    double customer_payment_total,
    double sale_amount_due,
    SALE_TRANSACTION *sale_transaction );

/* Process */
char *sale_update_transaction_date_time(
    SALE_TRANSACTION *sale_transaction );

char *sale_update_system_string(
    const char *SALE_TABLE );
```

SALE (3)

```
/* Driver */
void sale_update(
    SALE_TABLE,
    full_name,
    street_address,
    sale_date_time,
    sale_trigger_new()->
       sale_fetch->
       inventory_sale_boolean,
    sale_trigger_new()->
       sale_fetch->
       specific_inventory_sale_boolean,
    sale_trigger_new()->
       sale_fetch->
       fixed_service_sale_boolean,
    sale_trigger_new()->
       sale_fetch->
       hourly_service_sale_boolean,
    sale_trigger_new()->inventory_sale_total,
    sale_trigger_new()->specific_inventory_sale_total,
    sale_trigger_new()->fixed_service_sale_total,
    sale_trigger_new()->hourly_service_sale_total,
    sale_trigger_new()->gross_revenue,
    sale_trigger_new()->sales_tax,
    sale_trigger_new()->invoice_amount,
    sale_trigger_new()->customer_payment_total,
    sale_trigger_new()->amount_due,
    sale_trigger_new()->sale_transaction );

if ( sale_trigger_new()->sale_transaction )
{
    void subsidiary_transaction_execute(
        application_name,
        sale_trigger_new()->
          sale_transaction->
          subsidiary_transaction->
          delete_transaction,
        sale_trigger_new()->
          sale_transaction->
          subsidiary_transaction->
          insert_transaction,
        sale_trigger_new()->
          sale_transaction->
          subsidiary_transaction->
          update_template );
}
```

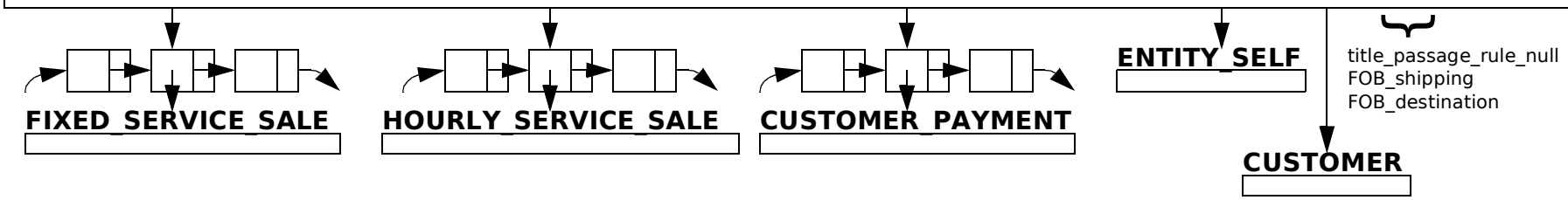
SALE_FETCH (1)

```
/* Usage */
SALE_FETCH *sale_fetch_new(
    const char *SALE_SELECT,
    const char *SALE_TABLE,
    char *full_name,
    char *street_address,
    char *sale_date_time );

/* Process */
SALE_FETCH *sale_fetch_calloc(
    void );

FOLDER *folder_fetch(
    SALE_TABLE /* folder_name */,
    (char *)0 /* role_name */,
    (LIST *)0 /* role_attribute_exclude_name_list */,
    1 /* fetch_folder_attribute_list */,
    0 /* not fetch_attribute */ );

boolean sale_fetch_inventory_sale_boolean(
    LIST *folder_fetch()->folder_attribute_list );

boolean sale_fetch_specific_inventory_sale_boolean(
    LIST *folder_fetch()->folder_attribute_list );

boolean sale_fetch_title_passage_rule_boolean(
    LIST *folder_fetch()->folder_attribute_list );

boolean sale_fetch_shipping_charge_boolean(
    LIST *folder_fetch()->folder_attribute_list );

boolean sale_fetch_instructions_boolean(
    LIST *folder_fetch()->folder_attribute_list );

boolean sale_fetch_hourly_service_sale_boolean(
    LIST *folder_fetch()->folder_attribute_list );

boolean sale_fetch_fixed_service_sale_boolean(
    LIST *folder_fetch()->folder_attribute_list );

char *sale_fetch_select(
    const char *SALE_SELECT,
    boolean sale_fetch_inventory_sale_boolean(),
    boolean sale_fetch_specific_inventory_sale_boolean(),
    boolean sale_fetch_title_passage_rule_boolean(),
    boolean sale_fetch_shipping_charge_boolean(),
    boolean sale_fetch_instructions_boolean(),
    boolean sale_fetch_hourly_service_sale_boolean(),
    boolean sale_fetch_fixed_service_sale_boolean() );

static char *sale_primary_where(
    full_name,
    street_address,
    sale_date_time );

char *appaserver_system_string(
    sale_fetch_select(),
    SALE_TABLE,
    sale_primary_where() );

free( sale_fetch_select() );

char *string_pipe_fetch(
    appaserver_system_string() );

/* May be deleted */
if ( !string_pipe_fetch() ) return null;

free( appaserver_system_string() );

SALE_FETCH *sale_fetch_parse(
    full_name,
    street_address,
    sale_date_time,
    sale_fetch_calloc() /* in/out */,
    sale_fetch_inventory_sale_boolean(),
    sale_fetch_specific_inventory_sale_boolean(),
    sale_fetch_title_passage_rule_boolean(),
    sale_fetch_shipping_charge_boolean(),
    sale_fetch_instructions_boolean(),
    sale_fetch_hourly_service_sale_boolean(),
    sale_fetch_fixed_service_sale_boolean(),
    string_pipe_fetch() );

if ( sale_fetch_inventory_sale_boolean() )
{
    LIST *inventory_sale_list(
        INVENTORY_SALE_TABLE,
        full_name,
        street_address,
        sale_date_time );
}

if ( sale_fetch_specific_inventory_sale_boolean() )
{
    LIST *specific_inventory_sale_list(
        SPECIFIC_INVENTORY_SALE_TABLE,
        full_name,
        street_address,
        sale_date_time );
}

if (   sale_fetch_inventory_sale_boolean()
||     sale_fetch_specific_inventory_sale_boolean() )
{
    CUSTOMER *customer_fetch(
        full_name,
        street_address,
        0 /* not fetch_entity_boolean */,
        0 /* not fetch_payable_balance_boolean */ );

    if ( !customer_fetch()->sales_tax_exempt_boolean )
    {
        ENTITY_SELF *entity_self_fetch(
            0 /* not fetch_entity_boolean */ );

        double self_tax_state_sales_tax_rate(
            entity_self_fetch()->full_name,
            entity_self_fetch()->street_address );
    }
}
```



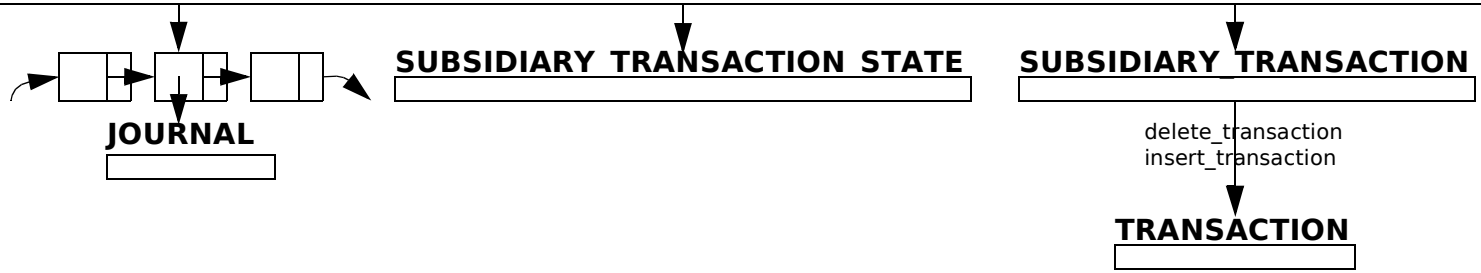
SALE_FETCH (2)

```
/* Process (continued) */
if ( sale_fetch_fixed_service_sale_boolean() )
{
   LIST *fixed_service_sale_list(
      FIXED_SERVICE_TABLE,
      full_name,
      street_address,
      sale_date_time );
}

if ( sale_fetch_hourly_service_sale_boolean() )
{
   LIST *hourly_service_sale_list(
      HOURLY_SERVICE_TABLE,
      full_name,
      street_address,
      sale_date_time );
}

LIST *customer_payment_list(
   CUSTOMER_PAYMENT_TABLE,
   full_name,
   street_address,
   sale_date_time );

/* Usage */
SALE_FETCH *sale_fetch_parse(
   char *full_name,
   char *street_address,
   char *sale_date_time,
   SALE_FETCH *sale_fetch_calloc /* in/out */,
   boolean inventory_sale_boolean,
   boolean specific_inventory_sale_boolean,
   boolean title_passage_rule_boolean,
   boolean shipping_charge_boolean,
   boolean instructions_boolean,
   boolean hourly_service_sale_boolean,
   boolean fixed_service_sale_boolean,
   char *string_pipe_fetch );

/* Process */
if ( title_passage_rule_boolean )
{
   enum predictive_title_passage_rule
      predictive_resolve_title_passage_rule(
         title_passage_rule_string );
}

/* Attributes */
char *full_name;
char *street_address;
char *sale_date_time;
FOLDER *folder_fetch;
boolean inventory_sale_boolean;
boolean specific_inventory_sale_boolean;
boolean title_passage_rule_boolean;
boolean shipping_charge_boolean;
boolean instructions_boolean;
boolean hourly_service_sale_boolean;
boolean fixed_service_sale_boolean;
double shipping_charge;
char *instructions;
double gross_revenue;
double invoice_amount;
double payment_total;
double amount_due;
char *completed_date_time;
char *shipped_date_time;
char *arrived_date;
char *uncollectible_date_time;
char *transaction_date_time;
double inventory_sale_total;
double specific_inventory_sale_total;
double sales_tax;
char *title_passage_rule_string;
double fixed_service_sale_total;
double hourly_service_sale_total;
enum predictive_title_passage_rule
   predictive_title_passage_rule;
LIST *inventory_sale_list;
LIST *specific_inventory_sale_list;
CUSTOMER*customer;
ENTITY_SELF *entity_self;
double self_tax_state_sales_tax_rate;
LIST *fixed_service_sale_list;
LIST *hourly_service_sale_list;
LIST *customer_payment_list;
```



SALE_TRANSACTION (1)

```
/* Usage */
SALE_TRANSACTION *sale_transaction_new(
    char *full_name,
    char *street_address,
    char *state,
    char *preupdate_full_name,
    char *preupdate_street_address,
    enum predictive_title_passage_rule
      predictive_title_passage_rule,
    char *completed_date_time,
    char *shipped_date_time,
    char *arrived_date,
    char *prior_transaction_date_time,
    double shipping_charge,
    double inventory_sale_total,
    double inventory_sale_CGS_total,
    double specific_inventory_sale_total,
    double specific_inventory_sale_CGS_total,
    double sale_gross_revenue,
    double sale_sales_tax,
    double sale_invoice_amount );

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

SALE_TRANSACTION *sale_transaction_calloc(
    void );

char *sale_transaction_date_time(
    enum predictive_title_passage_rule
      predictive_title_passage_rule,
    char *completed_date_time,
    char *shipped_date_time,
    char *arrived_date );

if ( sale_transaction_date_time() )
{
    LIST *sale_transaction_journal_list(
      shipping_charge,
      inventory_sale_total,
      inventory_sale_CGS_total,
      specific_inventory_sale_total,
      specific_inventory_sale_CGS_total,
      sale_invoice_amount,
      sale_gross_revenue,
      sale_sales_tax );
}

SUBSIDIARY_TRANSACTION_STATE *
    subsidiary_transaction_state_new(
      "preupdate_full_name"
        /* preupdate_full_name_placeholder */,
      "preupdate_street_address"
        /* preupdate_street_address_placeholder */,
      "preupdate_transaction_date_time"
        /* preupdate_foreign_date_time_placeholder */,
      state,
      preupdate_full_name,
      preupdate_street_address,
      prior_transaction_date_time
        /* preupdate_foreign_date_time */,
      full_name,
      street_address,
      sale_transaction_date_time()
        /* foreign_date_time */,
      sale_transaction_journal_list()
        /* insert_journal_list */ );

SUBSIDIARY_TRANSACTION *
    subsidiary_transaction_new(
      SALE_TABLE
        /* foreign_table_name */,
      "full_name"
        /* foreign_full_name_column */,
      "street_address"
        /* foreign_street_address_column */,
      "transaction_date_time"
        /* foreign_date_time_column */,
      prior_transaction_date_time,
      sale_transaction_journal_list
        /* insert_journal_list */,
      sale_invoice_amount
        /* foreign_amount */,
      "Sale" /* transaction_memo */,
      subsidiary_transaction_state_new()->
        subsidiary_transaction_insert,
      subsidiary_transaction_state_new()->
        subsidiary_transaction_delete );

/* Attributes */
char *transaction_date_time;
LIST *journal_list;
SUBSIDIARY_TRANSACTION_STATE *
    subsidiary_transaction_state;
SUBSIDIARY_TRANSACTION *subsidiary_transaction;
```

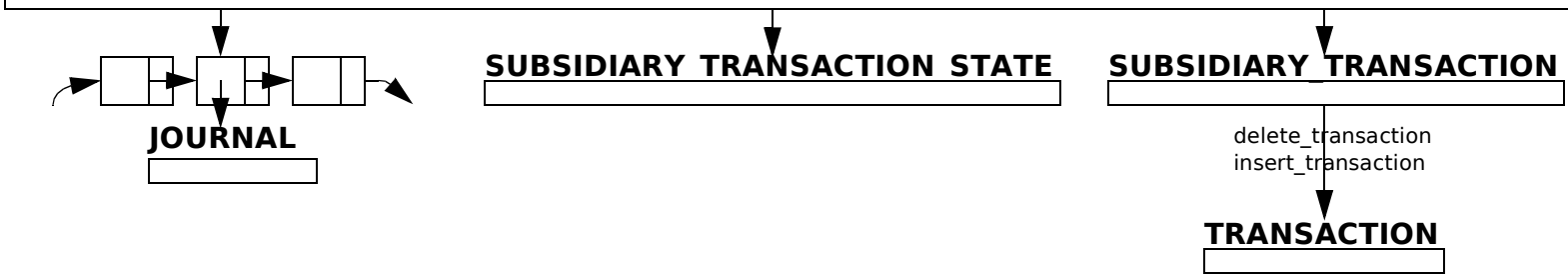


SALE_TRANSACTION (2)

```
/* Usage */
LIST *sale_transaction_journal_list(
    double shipping_charge,
    double inventory_sale_total,
    double inventory_sale_CGS_total,
    double specific_inventory_sale_total,
    double specific_inventory_sale_CGS_total,
    double sale_invoice_amount,
    double sale_gross_revenue,
    double sale_sales_tax );

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

double sale_transaction_debit_sum(
    double sale_invoice_amount,
    double inventory_sale_CGS_total,
    double specific_inventory_sale_CGS_total );

double sale_transaction_credit_sum(
    double shipping_charge,
    double inventory_sale_total,
    double specific_inventory_sale_total,
    double sale_gross_revenue,
    double sale_sales_tax );

double sale_transaction_difference(
    double sale_transaction_debit_sum(),
    double sale_transaction_credit_sum() );

if ( sale_transaction_difference() ) exit( 1 );

ACCOUNT *account_receivable(
    ACCOUNT_RECEIVABLE_KEY,
    __FUNCTION__ );

JOURNAL *journal_account_new(
    sale_invoice_amount,
    account_receivable() /* debit_account */,
    (ACCOUNT *)0 /* credit_account */ );

list_set( list, journal_account_new() );

if (    inventory_sale_total
||      specific_inventory_sale_total )
{
    double sale_transaction_CGS_total(
        double inventory_sale_CGS_total,
        double specific_inventory_sale_CGS_total );

    ACCOUNT *account_cost_of_goods_sold(
        ACCOUNT_CGS_KEY,
        __FUNCTION__ );

    JOURNAL *journal_account_new(
        sale_transaction_CGS_total()
            /* journal_amount */,
        account_cost_of_goods_sold()
            /* debit_account */,
        (ACCOUNT *)0
            /* credit_account */ );

    list_set( list, journal_account_new() );

    double sale_transaction_inventory_total(
        double inventory_sale_total,
        double specific_inventory_sale_total );

    ACCOUNT *account_inventory(
        ACCOUNT_INVENTORY_KEY,
        _FUNCTION__ );

    JOURNAL *journal_account_new(
        sale_transaction_inventory_total()
            /* journal_amount */,
        (ACCOUNT *)0
            /* debit_account */,
        account_inventory()
            /* credit_account */ );

    list_set( list, journal_account_new() );
}

ACCOUNT *account_revenue(
    ACCOUNT_REVENUE_KEY,
    __FUNCTION__ );

JOURNAL *journal_account_new(
    sale_gross_revenue /* journal_amount */,
    (ACCOUNT *)0 /* debit_account */,
    account_revenue() /* credit_account */ );

list_set( list, journal_account_new() );

if ( shipping_charge )
{
    ACCOUNT *account_shipping_revenue(
        ACCOUNT_SHIPPING_REVENUE_KEY,
        __FUNCTION__ );

    JOURNAL *journal_account_new(
        shipping_charge /* journal_amount */,
        (ACCOUNT *)0 /* debit_account */,
        account_shipping_revenue() /* credit_account */ );

    list_set( list, journal_account_new() );
}

if ( sale_sales_tax )
{
    ACCOUNT *account_sales_tax_payable(
        ACCOUNT_SALES_TAX_PAYABLE_KEY,
        __FUNCTION__ );

    JOURNAL *journal_account_new(
        sale_sales_tax /* journal_amount */,
        (ACCOUNT *)0 /* debit_account */,
        account_sales_tax_payable() /* credit_account */ );

    list_set( list, journal_account_new() );
}
```

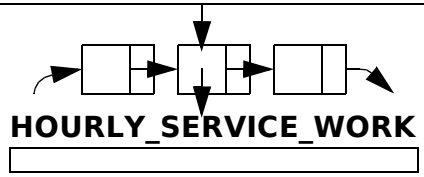
SALE_LOSS_TRANSACTION

```
/* Usage */
SALE_LOSS_TRANSACTION *sale_loss_transaction_new(
    char *full_name,
    char *street_address,
    char *uncollectible_date_time,
    char *state,
    char *preupdate_full_name,
    char *preupdate_street_address,
    char *preupdate_uncollectible_date_time,
    double sale_amount_due );

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

SALE_LOSS_TRANSACTION *sale_loss_transaction_calloc(
    void );

if ( uncollectible_date_time )
{
    char *account_loss_string(
       ACCOUNT_LOSS_KEY,
       __FUNCTION__ );

    ACCOUNT *debit_account =
       account_fetch(
          account_loss_string(),
          1 /* fetch_subclassification */,
          1 /* fetch_element */ );

    char *account_receivable_string(
       ACCOUNT_RECEIVABLE_KEY,
       __FUNCTION__ );

    ACCOUNT *credit_account =
       account_fetch(
          account_receivable_string(),
          1 /* fetch_subclassification */,
          1 /* fetch_element */ );

    LIST *journal_binary_list(
       (char *)0 /* full_name */,
       (char *)0 /* street_address */,
       (char *)0 /* transaction_date_time */,
       sale_amount_due
          /* transaction_amount */,
       debit_account,
       credit_account );
}

SUBSIDIARY_TRANSACTION_STATE *
    subsidiary_transaction_state_new(
       "preupdate_full_name"
          /* preupdate_full_name_placeholder */,
       "preupdate_street_address"
          /* preupdate_street_address_placeholder */,
       "preupdate_uncollectible_date_time"
          /* preupdate_foreign_date_time_placeholder */,
       state,
       preupdate_full_name,
       preupdate_street_address,
       preupdate_uncollectible_date_time
          /* preupdate_foreign_date_time */,
       full_name,
       street_address,
       uncollectible_date_time
          /* foreign_date_time */,
       journal_binary_list()
          /* insert_journal_list */ );

SUBSIDIARY_TRANSACTION *
    subsidiary_transaction_new(
       SALE_TABLE
          /* foreign_table_name */,
       "full_name"
          /* foreign_full_name_column */,
       "street_address"
          /* foreign_street_address_column */,
       "uncollectible_date_time"
          /* foreign_date_time_column */,
       uncollectible_date_time
          /* prior_transaction_date_time */,
       journal_binary_list()
          /* insert_journal_list */,
       sale_amount_due
          /* foreign_amount */,
       "Uncollectible loss" /* transaction_memo */,
       subsidiary_transaction_state_new()->
          subsidiary_transaction_insert,
       subsidiary_transaction_state_new()->
          subsidiary_transaction_delete );

/* Attributes */
char *account_loss_string;
ACCOUNT *debit_account;
char *account_receivable_string;
ACCOUNT *credit_account;
LIST *journal_binary_list;
SUBSIDIARY_TRANSACTION_STATE *
    subsidiary_transaction_state;
SUBSIDIARY_TRANSACTION *subsidiary_transaction;
```



HOURLY_SERVICE_SALE (1)

```
/* Usage */
LIST *hourly_service_sale_list(
    const char *HOURLY_SERVICE_SALE_SELECT,
    const char *HOURLY_SERVICE_SALE_TABLE,
    char *full_name,
    char *street_address,
    char *sale_date_time );

/* Process */
static char *sale_primary_where(
    full_name,
    street_address,
    sale_date_time );

char *appaserver_system_string(
    HOURLY_SERVICE_SALE_SELECT,
    HOURLY_SERVICE_SALE_TABLE,
    sale_primary_where() );

FILE *appaserver_input_pipe(
    appaserver_system_string() );

free( appaserver_system_string() );

for char *string_input( appaserver_input_pipe() )
{
    HOURLY_SERVICE_SALE *hourly_service_sale_parse(
        full_name,
        street_address,
        sale_date_time,
        string_input() );

    list_set(
        list,
        hourly_service_sale_parse() );
}

/* Usage */
HOURLY_SERVICE_SALE *hourly_service_sale_parse(
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *string_input );

/* Process */
HOURLY_SERVICE_SALE *hourly_service_sale_new(
    full_name,
    street_address,
    sale_date_time,
    strdup( service_name ),
    strdup( service_description ) );

double hourly_service_sale_estimated_revenue(
    this->estimated_hours,
    this->hourly_rate,
    this->discount_amount );

LIST *hourly_service_work_list(
    HOURLY_SERVICE_WORK_SELECT,
    HOURLY_SERVICE_WORK_TABLE,
    full_name,
    street_address,
    sale_date_time,
    this->service_name,
    this->service_description );

double hourly_service_work_hours(
    hourly_service_work_list() );

double hourly_service_sale_net_revenue(
    hourly_service_work_hours(),
    this->hourly_rate,
    this->discount_amount );

/* Usage */
HOURLY_SERVICE_SALE *hourly_service_sale_new(
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *service_name,
    char *service_description );

/* Process */
HOURLY_SERVICE_SALE *hourly_service_sale_calloc(
    void );

/* Usage */
HOURLY_SERVICE_SALE *hourly_service_sale_fetch(
    const char *HOURLY_SERVICE_SALE_SELECT,
    const char *HOURLY_SERVICE_SALE_TABLE,
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *service_name,
    char *service_description );

/* Process */
static char *hourly_service_sale_primary_where(
    full_name,
    street_address,
    sale_date_time,
    service_name,
    service_description );

char *appaserver_system_string(
    HOURLY_SERVICE_SALE_SELECT,
    HOURLY_SERVICE_SALE_TABLE,
    hourly_service_sale_primary_where() );

char *string_fetch( appaserver_system_string() );

if ( !string_fetch() ) return null;

HOURLY_SERVICE_SALE *hourly_service_sale_parse(
    full_name,
    street_address,
    sale_date_time,
    string_fetch() );

/* Attributes */
char *full_name;
char *street_address;
char *sale_date_time;
char *service_name;
char *service_description;
double estimated_hours;
double hourly_rate;
double estimated_revenue;
double discount_amount;
double work_hours; /* from parse */
double net_revenue; /* from parse */
double hourly_service_sale_estimated_revenue;
LIST *hourly_service_work_list;
double hourly_service_work_hours; /* for update */
double hourly_service_sale_net_revenue; /* for update */
```



HOURLY_SERVICE_SALE (2)

```
/* Usage */
static char *hourly_service_sale_primary_where(
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *service_name,
    char *service_description );

/* Process */
static char *sale_primary_where(
    full_name,
    street_address,
    sale_date_time );

/* Usage */
double hourly_service_sale_estimated_revenue(
    double estimated_hours,
    double hourly_rate,
    double discount_amount );

/* Process */
(A * B) - C;

/* Usage */
double hourly_service_sale_net_revenue(
    double hourly_service_work_hours(),
    double hourly_rate,
    double discount_amount );

/* Process */
(A * B) - C;

/* Usage */
double hourly_service_sale_total(
    LIST *hourly_service_sale_list );

/* Process */
for HOURLY_SERVICE_SALE *hourly_service_sale in
    hourly_service_sale_list
{
    total +=
        hourly_service_sale->
            net_revenue;
}

/* Usage */
void hourly_service_sale_update(
    const char *HOURLY_SERVICE_SALE_TABLE,
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *service_name,
    char *service_description,
    double hourly_service_sale_estimated_revenue,
    double hourly_service_work_hours,
    double hourly_service_sale_net_revenue );

/* Process */
char *hourly_service_sale_update_system_string(
    const char *HOURLY_SERVICE_SALE_TABLE );

FILE *appaserver_output_pipe(
    hourly_service_sale_update_system_string() );

free( hourly_service_sale_update_system_string() );

fprintf(
    appaserver_output_pipe(),
    "%s^%s^%s^%s^%s^estimated_revenue^%.2lf\n",
    full_name,
    street_address,
    sale_date_time,
    service_name,
    service_description,
    hourly_service_sale_estimated_revenue );

fprintf(
    appaserver_output_pipe(),
    "%s^%s^%s^%s^%s^work_hours^%.2lf\n",
    full_name,
    street_address,
    sale_date_time,
    service_name,
    service_description,
    hourly_service_work_list_hours );

fprintf(
    appaserver_output_pipe(),
    "%s^%s^%s^%s^%s^net_revenue^%.2lf\n",
    full_name,
    street_address,
    sale_date_time,
    service_name,
    service_description,
    hourly_service_sale_net_revenue );

pclose( hourly_service_sale_update_open() );
```

HOURLY_SERVICE_SALE (3)

```
/* Driver */
void hourly_service_sale_trigger(
    char *application_name,
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *service_name,
    char *service_description,
    char *state );

/* Process */
if ( strcmp(
        state,
        APPASERVER_PREDELETE_STATE ) == 0 )
{
    return;
}

if (    strcmp(
            state,
            APPASERVER_INSERT_STATE ) == 0
||      strcmp(
            state,
            APPASERVER_UPDATE_STATE ) == 0 )
{
    HOURLY_SERVICE_SALE *hourly_service_sale_fetch(
        HOURLY_SERVICE_SALE_SELECT,
        HOURLY_SERVICE_SALE_TABLE,
        full_name,
        street_address,
        sale_date_time,
        service_name,
        service_description );

    void hourly_service_sale_update(
        HOURLY_SERVICE_SALE_TABLE,
        full_name,
        street_address,
        sale_date_time,
        service_name,
        service_description,
        hourly_service_sale_fetch()->
            hourly_service_sale_estimated_revenue,
        hourly_service_sale_fetch()->
            hourly_service_work_hours,
        hourly_service_sale_fetch()->net_revenue );
}

SALE *sale_trigger_new(
    full_name,
    street_address,
    sale_date_time,
    state,
    (char *)0 /* preupdate_full_name */,
    (char *)0 /* preupdate_street_address */,
    (char *)0 /* preupdate_uncollectible_date_time */ );

if ( !sale_trigger_new() ) return;

void sale_update(
    SALE_TABLE,
    full_name,
    street_address,
    sale_date_time,
    sale_trigger_new()->
        sale_fetch->
        inventory_sale_boolean,
    sale_trigger_new()->
        sale_fetch->
        specific_inventory_sale_boolean,
    sale_trigger_new()->
        sale_fetch->
        fixed_service_sale_boolean,
    sale_trigger_new()->
        sale_fetch->
        hourly_service_sale_boolean,
    sale_trigger_new()->inventory_sale_total,
    sale_trigger_new()->specific_inventory_sale_total,
    sale_trigger_new()->fixed_service_sale_total,
    sale_trigger_new()->hourly_service_sale_total,
    sale_trigger_new()->gross_revenue,
    sale_trigger_new()->sales_tax,
    sale_trigger_new()->invoice_amount,
    sale_trigger_new()->customer_payment_total,
    sale_trigger_new()->amount_due,
    sale_trigger_new()->sale_transaction );

if ( sale_trigger_new()->sale_transaction )
{
    void subsidiary_transaction_execute(
        application_name,
        sale_trigger_new()->
            sale_transaction->
            subsidiary_transaction->
            delete_transaction,
        sale_trigger_new()->
            sale_transaction->
            subsidiary_transaction->
            insert_transaction,
        sale_trigger_new()->
            sale_transaction->
            subsidiary_transaction->
            update_template );
}
```

HOURLY_SERVICE_WORK (1)

```
/* Usage */
LIST *hourly_service_work_list(
    const char *HOURLY_SERVICE_WORK_SELECT,
    const char *HOURLY_SERVICE_WORK_TABLE,
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *service_name,
    char *service_description );

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

static char *hourly_service_sale_primary_where(
    full_name,
    street_address,
    sale_date_time,
    service_name,
    service_description );

char *appaserver_system_string(
    HOURLY_SERVICE_WORK_SELECT,
    HOURLY_SERVICE_WORK_TABLE,
    hourly_service_sale_primary_where() );

FILE *appaserver_input_pipe(
    appaserver_system_string() );

free( appaserver_system_string() );

for char *string_input( appaserver_input_pipe() )
{
    HOURLY_SERVICE_WORK *hourly_service_work_parse(
        full_name,
        street_address,
        sale_date_time,
        service_name,
        service_description,
        string_input() );

    list_set(
        list,
        hourly_service_work_parse() );
}

/* Usage */
HOURLY_SERVICE_WORK *hourly_service_work_parse(
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *service_name,
    char *service_description,
    char *string_input );

/* Process */
HOURLY_SERVICE_WORK *hourly_service_work_new(
    full_name,
    street_address,
    sale_date_time,
    service_name,
    service_description,
    strdup( begin_work_date_time[] ) );

double sale_work_hours(
    this->begin_work_date_time,
    this->end_work_date_time );

/* Usage */
HOURLY_SERVICE_WORK *hourly_service_work_new(
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *service_name,
    char *service_description,
    char *begin_work_date_time );

/* Process */
HOURLY_SERVICE_WORK *hourly_service_work_calloc(
    void );

/* Usage */
HOURLY_SERVICE_WORK *hourly_service_work_fetch(
    const char *HOURLY_SERVICE_WORK_SELECT,
    const char *HOURLY_SERVICE_WORK_TABLE,
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *service_name,
    char *service_description,
    char *begin_work_date_time );

/* Process */
static char *hourly_service_work_primary_where(
    full_name,
    street_address,
    sale_date_time,
    service_name,
    service_description,
    begin_work_date_time );

char *appaserver_system_string(
    HOURLY_SERVICE_WORK_SELECT,
    HOURLY_SERVICE_WORK_TABLE,
    hourly_service_work_primary_where() );

char *string_fetch( appaserver_system_string() );

if ( !string_fetch() ) return null;

HOURLY_SERVICE_WORK *hourly_service_work_parse(
    full_name,
    street_address,
    sale_date_time,
    service_name,
    service_description,
    string_fetch() /* string_input */ );

/* Attributes */
char *full_name;
char *street_address;
char *sale_date_time;
char *service_name;
char *service_description;
char *begin_work_date_time;
char *end_work_date_time;
char *work_description;
char *activity;
char *appaserver_full_name;
char *appaserver_street_address;
double work_hours; /* from parse */
double sale_work_hours; /* for update */
```

HOURLY_SERVICE_WORK (2)

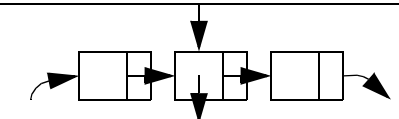
```
/* Usage */
static char *hourly_service_work_primary_where(
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *service_name,
    char *service_description,
    char *begin_work_date_time );

/* Process */
static char *hourly_service_sale_primary_where(
    full_name,
    street_address,
    sale_date_time,
    service_name,
    service_description );

/* Usage */
double hourly_service_work_hours(
    LIST *hourly_service_work_list );

/* Process */
for HOURLY_SERVICE_WORK *hourly_service work in
    hourly_service_work_list
{
    hours += hourly_service_work->work_hours;
}

/* Usage */
void hourly_service_work_update(
    const char *HOURLY_SERVICE_WORK_TABLE,
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *service_name,
    char *service_description,
    char *begin_work_date_time,
    double sale_work_hours );

/* Process */

char *hourly_service_work_update_system_string(
    const char *HOURLY_SERVICE_WORK_TABLE );

FILE *appaserver_output_pipe(
    hourly_service_work_update_system_string() );

free( hourly_service_work_update_system_string() );

fprintf(
    appaserver_output_pipe(),
    "%s^%s^%s^%s^%s^%s^work_hours^%.2lf\n",
    full_name,
    street_address,
    sale_date_time,
    service_name,
    service_description,
    begin_work_date_time,
    sale_work_hours );

pclose( appaserver_output_pipe() );
```

HOURLY_SERVICE_WORK (3)

```
/* Driver */
void hourly_service_work_trigger(
    char *application_name,
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *service_name,
    char *service_description,
    char *begin_work_date_time,
    char *state );

/* Process */
if ( strcmp(
       state,
       APPASERVER_PREDELETE_STATE ) == 0 )
{
    return;
}

if (    strcmp(
          state,
          APPASERVER_UPDATE_STATE ) == 0 )
{
    HOURLY_SERVICE_WORK *hourly_service_work_fetch(
        HOURLY_SERVICE_WORK_SELECT,
        HOURLY_SERVICE_WORK_TABLE,
        full_name,
        street_address,
        sale_date_time,
        service_name,
        service_description,
        begin_work_date_time );

    void hourly_service_work_update(
        HOURLY_SERVICE_WORK_TABLE,
        full_name,
        street_address,
        sale_date_time,
        service_name,
        service_description,
        begin_work_date_time,
        hourly_service_work_fetch()->sale_work_hours );
}

if (    strcmp(
          state,
          APPASERVER_UPDATE_STATE ) == 0
||      strcmp(
          state,
          APPASERVER_DELETE_STATE ) == 0 )
{
    HOURLY_SERVICE_SALE *hourly_service_sale_fetch(
        HOURLY_SERVICE_SALE_SELECT,
        HOURLY_SERVICE_SALE_TABLE,
        full_name,
        street_address,
        sale_date_time,
        service_name,
        service_description );

    if ( hourly_service_sale_fetch() )
    {
        hourly_service_sale_update(
            HOURLY_SERVICE_SALE_TABLE,
            full_name,
            street_address,
            sale_date_time,
            service_name,
            service_description,
            hourly_service_sale_fetch()->
              hourly_service_sale_estimated_revenue,
            hourly_service_sale_fetch()->
              hourly_service_work_hours,
            hourly_service_sale_fetch()->
              hourly_service_sale_net_revenue );
    }

    SALE *sale_trigger_new(
        full_name,
        street_address,
        sale_date_time,
        state,
        (char *)0 /* preupdate_full_name */,
        (char *)0 /* preupdate_street_address */,
        (char *)0 /* preupdate_uncollectible_date_time */ );

    if ( !sale_trigger_new() ) return;

    void sale_update(
        SALE_TABLE,
        full_name,
        street_address,
        sale_date_time,
        sale_trigger_new()->
          sale_fetch->
          inventory_sale_boolean,
        sale_trigger_new()->
          sale_fetch->
          specific_inventory_sale_boolean,
        sale_trigger_new()->
          sale_fetch->
          fixed_service_sale_boolean,
        sale_trigger_new()->
          sale_fetch->
          hourly_service_sale_boolean,
        sale_trigger_new()->inventory_sale_total,
        sale_trigger_new()->specific_inventory_sale_total,
        sale_trigger_new()->fixed_service_sale_total,
        sale_trigger_new()->hourly_service_sale_total,
        sale_trigger_new()->gross_revenue,
        sale_trigger_new()->sales_tax,
        sale_trigger_new()->invoice_amount,
        sale_trigger_new()->customer_payment_total,
        sale_trigger_new()->amount_due,
        sale_trigger_new()->sale_transaction );

    if ( sale_trigger_new()->sale_transaction )
    {
        void subsidiary_transaction_execute(
            application_name,
            sale_trigger_new()->
              sale_transaction->
              subsidiary_transaction->
              delete_transaction,
            sale_trigger_new()->
              sale_transaction->
              subsidiary_transaction->
              insert_transaction,
            sale_trigger_new()->
              sale_transaction->
              subsidiary_transaction->
              update_template );
    }
}
```

FIXED_SERVICE_SALE (1)

```
/* Usage */
LIST *fixed_service_sale_list(
    const char *FIXED_SERVICE_SALE_SELECT,
    const char *FIXED_SERVICE_SALE_TABLE,
    char *full_name,
    char *street_address,
    char *sale_date_time );

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

static char *sale_primary_where(
    full_name,
    street_address,
    sale_date_time );

char *appaserver_system_string(
    FIXED_SERVICE_SALE_SELECT,
    FIXED_SERVICE_SALE_TABLE,
    sale_primary_where() );

FILE *appaserver_input_pipe(
    appaserver_system_string() );

free( appaserver_system_string() );

while ( string_input( appaserver_input_pipe() ) )
{
    FIXED_SERVICE_SALE *hourly_service_sale_parse(
        full_name,
        street_address,
        sale_date_time,
        string_input() );

    list_set(
        list,
        fixed_service_sale_parse() );
}

/* Usage */
FIXED_SERVICE_SALE *fixed_service_sale_parse(
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *string_input );

/* Process */
FIXED_SERVICE_SALE *hourly_service_sale_new(
    full_name,
    street_address,
    sale_date_time,
    strdup( service_name[] ) );

LIST *fixed_service_work_list(
    FIXED_SERVICE_WORK_SELECT,
    FIXED_SERVICE_WORK_TABLE,
    full_name,
    street_address,
    sale_date_time,
    this->service_name );

double fixed_service_work_hours(
    fixed_service_work_list() );

double fixed_service_sale_net_revenue(
    this->fixed_price,
    this->discount_amount );

/* Usage */
FIXED_SERVICE_SALE *fixed_service_sale_new(
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *service_name );

/* Process */
FIXED_SERVICE_SALE *fixed_service_sale_calloc(
    void );

/* Usage */
FIXED_SERVICE_SALE *fixed_service_sale_fetch(
    const char *FIXED_SERVICE_SALE_SELECT,
    const char *FIXED_SERVICE_SALE_TABLE,
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *service_name );

/* Process */
static char *fixed_service_sale_primary_where(
    full_name,
    street_address,
    sale_date_time,
    service_name );

char *appaserver_system_string(
    FIXED_SERVICE_SALE_SELECT,
    FIXED_SERVICE_SALE_TABLE,
    fixed_service_sale_primary_where() );

char *string_fetch( appaserver_system_string() );

if ( !string_fetch() ) return null;

FIXED_SERVICE_SALE *fixed_service_sale_parse(
    full_name,
    street_address,
    sale_date_time,
    string_fetch() );

/* Attributes */
char *full_name;
char *street_address;
char *sale_date_time;
char *service_name;
double fixed_price;
double estimated_hours;
double discount_amount;
double work_hours; /* from parse */
double net_revenue; /* from parse */
LIST *fixed_service_work_list;
double fixed_service_work_hours; /* for update */
double fixed_service_sale_net_revenue; /* for update */
```



FIXED_SERVICE_WORK

FIXED_SERVICE_SALE (2)

```
/* Usage */
static char *fixed_service_sale_primary_where(
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *service_name );

/* Process */
static char *sale_primary_where(
    full_name,
    street_address,
    sale_date_time );

/* Usage */
double fixed_service_sale_net_revenue(
    double fixed_price,
    double discount_amount );

/* Process */
A - C;

/* Usage */
double fixed_service_sale_total(
    LIST *fixed_service_sale_list );

/* Process */
for FIXED_SERVICE_SALE *fixed_service_sale in
    fixed_service_sale_list
{
    total +=
        fixed_service_sale->
            net_revenue;
}

/* Usage */
void fixed_service_sale_update(
    const char *FIXED_SERVICE_SALE_TABLE,
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *service_name,
    double fixed_service_work_hours,
    double fixed_service_sale_net_revenue );

/* Process */
char *fixed_service_sale_update_system_string(
    const char *FIXED_SERVICE_SALE_TABLE );

FILE *appaserver_output_pipe(
    hourly_service_sale_update_system_string() );

free( fixed_service_sale_update_system_string() );

fprintf(
    appaserver_output_pipe(),
    "%s^%s^%s^%s^work_hours^%.2lf\n",
    full_name,
    street_address,
    sale_date_time,
    service_name,
    fixed_service_work_hours );

fprintf(
    appaserver_output_pipe(),
    "%s^%s^%s^%s^net_revenue^%.2lf\n",
    full_name,
    street_address,
    sale_date_time,
    service_name,
    fixed_service_sale_net_revenue );

pclose( appaserver_output_pipe() );
```

FIXED_SERVICE_SALE (3)

```
/* Driver */
void fixed_service_sale_trigger(
    char *application_name,
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *service_name,
    char *state );

/* Process */
if ( strcmp(
        state,
        APPASERVER_PREDELETE_STATE ) == 0 )
{
    return;
}

if (    strcmp(
            state,
            APPASERVER_INSERT_STATE ) == 0
||      strcmp(
            state,
            APPASERVER_UPDATE_STATE ) == 0 )
{
    FIXED_SERVICE_SALE *fixed_service_sale_fetch(
        FIXED_SERVICE_SALE_SELECT,
        FIXED_SERVICE_SALE_TABLE,
        full_name,
        street_address,
        sale_date_time,
        service_name );

    void fixed_service_sale_update(
        HOURLY_SERVICE_SALE_TABLE,
        full_name,
        street_address,
        sale_date_time,
        service_name,
        fixed_service_sale_fetch()->
            fixed_service_work_hours,
        fixed_service_sale_fetch()->net_revenue );
}

SALE *sale_trigger_new(
    full_name,
    street_address,
    sale_date_time,
    state,
    (char *)0 /* preupdate_full_name */,
    (char *)0 /* preupdate_street_address */,
    (char *)0 /* preupdate_uncollectible_date_time */ );

if ( !sale_trigger_new() ) return;

void sale_update(
    SALE_TABLE,
    full_name,
    street_address,
    sale_date_time,
    sale_trigger_new()->
        sale_fetch->
        inventory_sale_boolean,
    sale_trigger_new()->
        sale_fetch->
        specific_inventory_sale_boolean,
    sale_trigger_new()->
        sale_fetch->
        fixed_service_sale_boolean,
    sale_trigger_new()->
        sale_fetch->
        hourly_service_sale_boolean,
    sale_trigger_new()->inventory_sale_total,
    sale_trigger_new()->specific_inventory_sale_total,
    sale_trigger_new()->fixed_service_sale_total,
    sale_trigger_new()->hourly_service_sale_total,
    sale_trigger_new()->gross_revenue,
    sale_trigger_new()->sales_tax,
    sale_trigger_new()->invoice_amount,
    sale_trigger_new()->customer_payment_total,
    sale_trigger_new()->amount_due,
    sale_trigger_new()->sale_transaction );

if ( sale_trigger_new()->sale_transaction )
{
    void subsidiary_transaction_execute(
        application_name,
        sale_trigger_new()->
            sale_transaction->
            subsidiary_transaction->
            delete_transaction,
        sale_trigger_new()->
            sale_transaction->
            subsidiary_transaction->
            insert_transaction,
        sale_trigger_new()->
            sale_transaction->
            subsidiary_transaction->
            update_template );
}
```

FIXED_SERVICE_WORK (1)

```
/* Usage */
LIST *fixed_service_work_list(
    const char *FIXED_SERVICE_WORK_SELECT,
    const char *FIXED_SERVICE_WORK_TABLE,
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *service_name );

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

static char *fixed_service_work_where(
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *service_name );

char *appaserver_system_string(
    FIXED_SERVICE_WORK_SELECT,
    FIXED_SERVICE_WORK_TABLE,
    fixed_service_work_where() );

FILE *appaserver_input_pipe(
    appaserver_system_string() );

free( appaserver_system_string() );

for char *string_input( appaserver_input_pipe() )
{
    FIXED_SERVICE_WORK *fixed_service_work_parse(
        full_name,
        street_address,
        sale_date_time,
        service_name,
        string_input() );

    list_set(
        list,
        fixed_service_work_parse() );
}
```

```
/* Usage */
FIXED_SERVICE_WORK *fixed_service_work_parse(
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *service_name,
    char *string_input );

/* Process */
FIXED_SERVICE_WORK *fixed_service_work_new(
    full_name,
    street_address,
    sale_date_time,
    service_name,
    strdup( begin_work_date_time[] ) );

double sale_work_hours(
    this->begin_work_date_time,
    this->end_work_date_time );

/* Usage */
FIXED_SERVICE_WORK *fixed_service_work_new(
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *service_name,
    char *begin_work_date_time );

/* Process */
FIXED_SERVICE_WORK *fixed_service_work_calloc(
    void );

/* Usage */
FIXED_SERVICE_WORK *fixed_service_work_fetch(
    const char *FIXED_SERVICE_WORK_SELECT,
    const char *FIXED_SERVICE_WORK_TABLE,
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *service_name,
    char *begin_work_date_time );

/* Process */
static char *fixed_service_work_primary_where(
    full_name,
    street_address,
    sale_date_time,
    service_name,
    begin_work_date_time );

char *appaserver_system_string(
    FIXED_SERVICE_WORK_SELECT,
    FIXED_SERVICE_WORK_TABLE,
    fixed_service_work_primary_where() );

char *string_fetch( appaserver_system_string() );

if ( !string_fetch() ) return null;

FIXED_SERVICE_WORK *fixed_service_work_parse(
    full_name,
    street_address,
    sale_date_time,
    service_name,
    string_fetch() /* string_input */ );

/* Attributes */
char *full_name;
char *street_address;
char *sale_date_time;
char *service_name;
char *begin_work_date_time;
char *end_work_date_time;
char *work_description;
char *activity;
char *appaserver_full_name;
char *appaserver_street_address;
double work_hours; /* from parse */
double sale_work_hours; /* for update */
```

FIXED_SERVICE_WORK (2)

```
/* Usage */
static char *fixed_service_work_primary_where(
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *service_name,
    char *begin_work_date_time );

/* Process */
static char *fixed_service_sale_primary_where(
    full_name,
    street_address,
    sale_date_time,
    service_name );

/* Usage */
double fixed_service_work_hours(
    LIST *fixed_service_work_list );

/* Process */
for FIXED_SERVICE_WORK *fixed_service work in
    fixed_service_work_list
{
    hours += fixed_service_work->work_hours;
}

/* Usage */
void fixed_service_work_update(
    const char *FIXED_SERVICE_WORK_TABLE,
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *service_name,
    char *begin_work_date_time,
    double fixed_service_work_hours );

/* Process */
char *fixed_service_work_update_system_string(
    const char *FIXED_SERVICE_WORK_TABLE );

FILE *appaserver_output_pipe(
    fixed_service_work_update_system_string() );

free( fixed_service_work_update_system_string() );

fprintf(
    appaserver_output_pipe(),
    "%s^%s^%s^%s^%s^work_hours^%.2lf\n",
    full_name,
    street_address,
    sale_date_time,
    service_name,
    begin_work_date_time,
    fixed_service_work_hours );

pclose( appaserver_output_pipe() );
```

FIXED_SERVICE_WORK (3)

```
/* Driver */
void fixed_service_work_trigger(
    char *application_name,
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *service_name,
    char *begin_work_date_time,
    char *state );

/* Process */
if ( strcmp(
        state,
        APPASERVER_PREDELETE_STATE ) == 0 )
{
    return;
}

if (    strcmp(
            state,
            APPASERVER_INSERT_STATE ) == 0
||      strcmp(
            state,
            APPASERVER_UPDATE_STATE ) == 0 )
{
    FIXED_SERVICE_WORK *fixed_service_work_fetch(
        FIXED_SERVICE_WORK_SELECT,
        FIXED_SERVICE_WORK_TABLE,
        full_name,
        street_address,
        sale_date_time,
        service_name,
        begin_work_date_time );

    void fixed_service_work_update(
        FIXED_SERVICE_WORK_TABLE,
        full_name,
        street_address,
        sale_date_time,
        service_name,
        begin_work_date_time,
        hourly_service_work_fetch()->sale_work_hours );
}

SALE *sale_trigger_new(
    full_name,
    street_address,
    sale_date_time,
    state,
    (char *)0 /* preupdate_full_name */,
    (char *)0 /* preupdate_street_address */,
    (char *)0 /* preupdate_uncollectible_date_time */ );

if ( !sale_trigger_new() ) return;

void sale_update(
    SALE_TABLE,
    full_name,
    street_address,
    sale_date_time,
    sale_trigger_new()->
        sale_fetch->
        inventory_sale_boolean,
    sale_trigger_new()->
        sale_fetch->
        specific_inventory_sale_boolean,
    sale_trigger_new()->
        sale_fetch->
        fixed_service_sale_boolean,
    sale_trigger_new()->
        sale_fetch->
        hourly_service_sale_boolean,
    sale_trigger_new()->inventory_sale_total,
    sale_trigger_new()->specific_inventory_sale_total,
    sale_trigger_new()->fixed_service_sale_total,
    sale_trigger_new()->hourly_service_sale_total,
    sale_trigger_new()->gross_revenue,
    sale_trigger_new()->sales_tax,
    sale_trigger_new()->invoice_amount,
    sale_trigger_new()->customer_payment_total,
    sale_trigger_new()->amount_due,
    sale_trigger_new()->sale_transaction );

/* Shouldn't execute */
if ( sale_trigger_new()->sale_transaction )
{
    void subsidiary_transaction_execute(
        application_name,
        sale_trigger_new()->
            sale_transaction->
            subsidiary_transaction->
            delete_transaction,
        sale_trigger_new()->
            sale_transaction->
            subsidiary_transaction->
            insert_transaction,
        sale_trigger_new()->
            sale_transaction->
            subsidiary_transaction->
            update_template );
}
```

CUSTOMER_PAYMENT (1)

```
/* Usage */
LIST *customer_payment_list(
    const char *CUSTOMER_PAYMENT_SELECT,
    const char *CUSTOMER_PAYMENT_TABLE,
    char *full_name,
    char *street_address,
    char *sale_date_time );

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

static char *sale_primary_where(
    full_name,
    street_address,
    sale_date_time );

char *appaserver_system_string(
    CUSTOMER_PAYMENT_SELECT,
    CUSTOMER_PAYMENT_TABLE,
    sale_primary_where() );

FILE *appaserver_input_pipe(
    appaserver_system_string() );

free( appaserver_system_string() );

for char *string_input( appaserver_input_pipe() )
{
    CUSTOMER_PAYMENT *customer_payment_parse(
        full_name,
        street_address,
        sale_date_time,
        string_input() );

    list_set(
        list,
        customer_payment_parse() );
}

/* Usage */
CUSTOMER_PAYMENT *customer_payment_parse(
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *string_input );

/* Process */
CUSTOMER_PAYMENT *customer_payment_new(
    full_name,
    street_address,
    sale_date_time,
    strdup( payment_date_time[] ) );

/* Usage */
CUSTOMER_PAYMENT *customer_payment_new(
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *payment_date_time );

/* Process */
CUSTOMER_PAYMENT *customer_payment_calloc(
    void );

/* Attributes */
char *full_name;
char *street_address;
char *sale_date_time;
char *payment_date_time;
char *account;
double payment_amount;
int check_number;
CUSTOMER_PAYMENT_TRANSACTION *
    customer_payment_transaction;
```

CUSTOMER_PAYMENT (2)

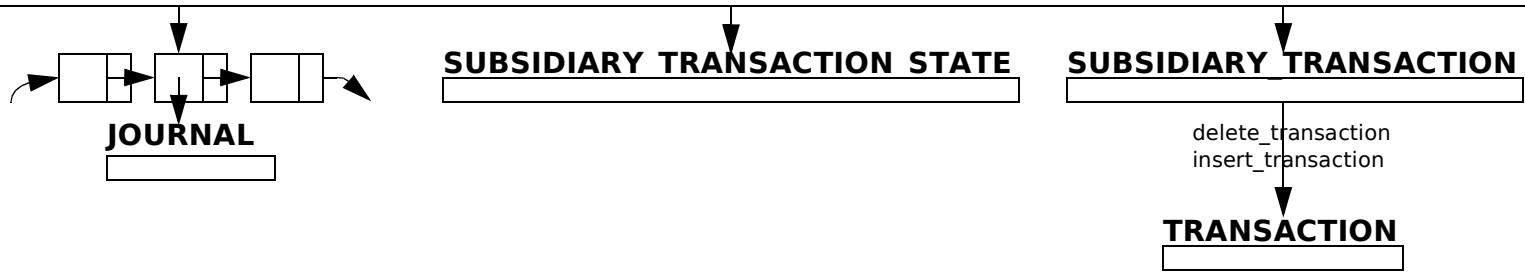
```
/* Usage */
CUSTOMER_PAYMENT *
   customer_payment_trigger_new(
      const char *CUSTOMER_PAYMENT_SELECT,
      const char *CUSTOMER_PAYMENT_TABLE,
      char *full_name,
      char *street_address,
      char *sale_date_time,
      char *payment_date_time,
      char *state,
      char *preupdate_full_name,
      char *preupdate_street_address,
      char *preupdate_payment_date_time );

/* Process */
CUSTOMER_PAYMENT *customer_payment_fetch(
   CUSTOMER_PAYMENT_SELECT,
   CUSTOMER_PAYMENT_TABLE,
   full_name,
   street_address,
   sale_date_time,
   payment_date_time );

if (   !customer_payment_fetch()->payment_amount
||     !customer_payment_fetch()->account )
{
   return this;
}

CUSTOMER_PAYMENT_TRANSACTION *
   customer_payment_transaction_new(
      full_name,
      street_address,
      payment_date_time,
      state,
      preupdate_full_name,
      preupdate_street_address,
      preupdate_payment_date_time,
      customer_payment_fetch()->
         account /* account_cash_string */,
      customer_payment_fetch()->
         payment_amount );

/* Usage */
CUSTOMER_PAYMENT *customer_payment_fetch(
   const char *CUSTOMER_PAYMENT_SELECT,
   const char *CUSTOMER_PAYMENT_TABLE,
   char *full_name,
   char *street_address,
   char *sale_date_time,
   char *payment_date_time );

/* Process */
static char *customer_payment_primary_where(
   full_name,
   street_address,
   sale_date_time,
   payment_date_time );

char *appaserver_system_string(
   CUSTOMER_PAYMENT_SELECT,
   CUSTOMER_PAYMENT_TABLE,
   customer_payment_primary_where() );

char *string_pipe_fetch(
   appaserver_system_string() );

free( appaserver_system_string() );

CUSTOMER_PAYMENT *customer_payment_parse(
   full_name,
   street_address,
   sale_date_time,
   string_pipe_fetch() /* string_input */ );

/* Usage */
static char *customer_payment_primary_where(
   char *full_name,
   char *street_address,
   char *sale_date_time,
   char *payment_date_time );

/* Process */
static char *sale_primary_where(
   full_name,
   street_address,
   sale_date_time );

/* Usage */
double customer_payment_total(
   LIST *customer_payment_list );

/* Process */
for CUSTOMER_PAYMENT *customer_payment in
   customer_payment_list
{
   total += customer_payment->payment_amount;
}
```

CUSTOMER_PAYMENT (3)

```
/* Driver */
void customer_payment_trigger(
    char *application_name,
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *payment_date_time,
    char *state,
    char *preupdate_full_name,
    char *preupdate_street_address,
    char *preupdate_payment_date_time );

/* Process */
if (    strcmp(
          state,
          APPASERVER_INSERT_STATE ) == 0
||      strcmp(
          state,
          APPASERVER_UPDATE_STATE ) == 0 )
{
    CUSTOMER_PAYMENT *
      customer_payment_trigger_new(
          CUSTOMER_PAYMENT_SELECT,
          CUSTOMER_PAYMENT_TABLE,
          full_name,
          street_address,
          sale_date_time,
          payment_date_time,
          state,
          preupdate_full_name,
          preupdate_street_address,
          preupdate_payment_date_time );

if ( customer_payment_trigger_new()->
       customer_payment_transaction )
{
    void subsidiary_transaction_execute(
          application_name,
          customer_payment_trigger_new()->
            customer_payment_transaction->
            subsidiary_transaction->
            delete_transaction,
          customer_payment_trigger_new()->
            customer_payment_transaction->
            subsidiary_transaction->
            insert_transaction,
          customer_payment_trigger_new()->
            customer_payment_transaction->
            subsidiary_transaction->
            update_template );
    }
}

SALE *sale_trigger_new(
    full_name,
    street_address,
    sale_date_time,
    state,
    (char *)0 /* preupdate_full_name */,
    (char *)0 /* preupdate_street_address */,
    (char *)0 /* preupdate_uncollectible_date_time */ );

if ( !sale_trigger_new() ) return;

void sale_update(
    SALE_TABLE,
    full_name,
    street_address,
    sale_date_time,
    sale_trigger_new()->
      sale_fetch->
      inventory_sale_boolean,
    sale_trigger_new()->
      sale_fetch->
      specific_inventory_sale_boolean,
    sale_trigger_new()->
      sale_fetch->
      fixed_service_sale_boolean,
    sale_trigger_new()->
      sale_fetch->
      hourly_service_sale_boolean,
    sale_trigger_new()->inventory_sale_total,
    sale_trigger_new()->specific_inventory_sale_total,
    sale_trigger_new()->fixed_service_sale_total,
    sale_trigger_new()->hourly_service_sale_total,
    sale_trigger_new()->gross_revenue,
    sale_trigger_new()->sales_tax,
    sale_trigger_new()->invoice_amount,
    sale_trigger_new()->customer_payment_total,
    sale_trigger_new()->amount_due,
    sale_trigger_new()->sale_transaction );

if ( sale_trigger_new()->sale_transaction )
{
    void subsidiary_transaction_execute(
      application_name,
      sale_trigger_new()->
        sale_transaction->
        subsidiary_transaction->
        delete_transaction,
      sale_trigger_new()->
        sale_transaction->
        subsidiary_transaction->
        insert_transaction,
      sale_trigger_new()->
        sale_transaction->
        subsidiary_transaction->
        update_template );
}
```

CUSTOMER_PAYMENT_TRANSACTION

```
/* Usage */
CUSTOMER_PAYMENT_TRANSACTION *
   customer_payment_transaction_new(
      char *full_name,
      char *street_address,
      char *payment_date_time,
      char *state,
      char *preupdate_full_name,
      char *preupdate_street_address,
      char *preupdate_payment_date_time,
      char *account_cash_string,
      double payment_amount );

/* Process */
if ( !payment_amount ) return exit( 1 );

CUSTOMER_PAYMENT_TRANSACTION *
   customer_payment_transaction_calloc(
      void );

ACCOUNT *debit_account =
   account_fetch(
      account_cash_string,
      1 /* fetch_subclassification */,
      1 /* fetch_element */ );

char *account_receivable_string(
   ACCOUNT_RECEIVABLE_KEY,
   __FUNCTION__ );

ACCOUNT *credit_account =
   account_fetch(
      account_receivable_string(),
      1 /* fetch_subclassification */,
      1 /* fetch_element */ );

LIST *journal_binary_list(
   (char *)0 /* full_name */,
   (char *)0 /* street_address */,
   (char *)0 /* transaction_date_time */,
   payment_amount
      /* transaction_amount */,
   debit_account,
   credit_account );

SUBSIDIARY_TRANSACTION_STATE *
   subsidiary_transaction_state_new(
      "preupdate_full_name"
         /* preupdate_full_name_placeholder */,
      "preupdate_street_address"
         /* preupdate_street_address_placeholder */,
      "preupdate_payment_date_time"
         /* preupdate_foreign_date_time_placeholder */,
      state,
      preupdate_full_name,
      preupdate_street_address,
      preupdate_payment_date_time
         /* preupdate_foreign_date_time */,
      full_name,
      street_address,
      payment_date_time
         /* foreign_date_time */,
      journal_binary_list()
         /* insert_journal_list */ );

SUBSIDIARY_TRANSACTION *
   subsidiary_transaction_new(
      CUSTOMER_PAYMENT_TABLE
         /* foreign_table_name */,
      "full_name"
         /* foreign_full_name_column */,
      "street_address"
         /* foreign_street_address_column */,
      "payment_date_time"
         /* foreign_date_time_column */,
      payment_date_time
         /* prior_transaction_date_time */,
      journal_binary_list()
         /* insert_journal_list */,
      payment_amount
         /* foreign_amount */,
      CUSTOMER_PAYMENT_MEMO
         /* transaction_memo */,
      subsidiary_transaction_state_new()->
         subsidiary_transaction_insert,
      subsidiary_transaction_state_new()->
         subsidiary_transaction_delete );

/* Attributes */
ACCOUNT *debit_account;
char *account_receivable_string;
ACCOUNT *credit_account;
LIST *journal_binary_list;
SUBSIDIARY_TRANSACTION_STATE *
   subsidiary_transaction_state;
SUBSIDIARY_TRANSACTION *subsidiary_transaction;
```



PURCHASE (1)

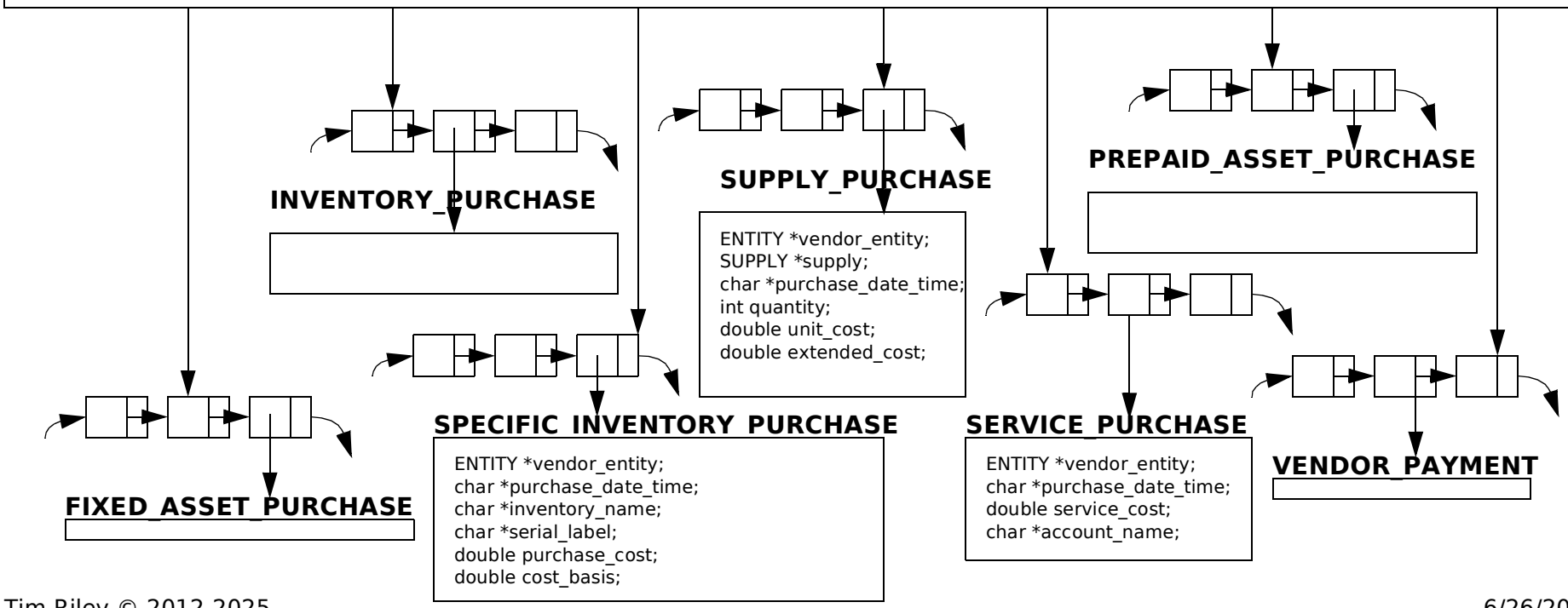
```
/* Usage */
PURCHASE *purchase_fetch(
    char *full_name,
    char *street_address,
    char *purchase_date_time );

/* Input */
ENTITY *vendor_entity =
    entity_new(
      full_name,
      street_address );

char *purchase_date_time;
double sales_tax;
double freight_in;
char *title_passage_rule_string;
char *shipped_date;
char *arrived_date_time;
char *program_name;
char *property_street_address;

/* Process */
enum title_passage_rule title_passage_rule =
    predictive_title_passage_rule_resolve(
      title_passage_rule_string );

char *purchase_primary_where(
    char *full_name,
    char *street_address,
    char *purchase_date_time );

char *purchase_system_string(
    char *where,
    char *order );

LIST *purchase_system_list(
    purchase_system_string() );

PURCHASE *purchase_parse(
    char *input );

LIST *fixed_asset_purchase_list =
    fixed_asset_purchase_list_fetch(
      vendor_entity->full_name,
      vendor_entity->street_address,
      purchase_date_time );

LIST *vendor_payment_list =
    vendor_payment_list_fetch(
      full_name,
      street_address,
      purchase_date_time );

PURCHASE *purchase_steady_state(
    sales_tax,
    freight_in,
    arrived_date,
    fixed_asset_purchase_list,
    inventory_purchase_list,
    specific_inventory_purchase_list,
    supply_purchase_list,
    prepaid_asset_purchase_list,
    inventory_purchase_return_list,
    vendor_payment_list,
    purchase );

double fixed_asset_purchase_total =
    fixed_asset_purchase_total(
      fixed_asset_purchase_list );

double inventory_purchase_total =
    inventory_purchase_total(
      inventory_purchase_list );

double specific_inventory_purchase_total =
    specific_inventory_purchase_total(
      specific_inventory_purchase_list );

double supply_purchase_total =
    supply_purchase_total(
      supply_purchase_list );

double prepaid_asset_purchase_total =
    prepaid_asset_purchase_total(
      prepaid_asset_purchase_list );

double inventory_purchase_return_total =
    inventory_purchase_return_total(
      inventory_purchase_return_list );

double vendor_payment_total =
    vendor_payment_total(
      vendor_payment_list );

double purchase_invoice_amount(
    fixed_asset_purchase_total,
    inventory_purchase_total,
    specific_inventory_purchase_total,
    supply_purchase_total,
    prepaid_asset_purchase_total,
    sales_tax,
    freight_in );

double purchase_amount_due(
      purchase_invoice_amount,
      vendor_payment_total )
{
    return A - B;
}

double purchase_asset_cost_basis(
          asset_cost,
          sales_tax,
          freight_in,
          fixed_asset_purchase_total(),
          inventory_purchase_total(),
          specific_inventory_purchase_total(),
          supply_expense_purchase_total(),
          prepaid_asset_purchase_total() );
```

ENTITY

FOB_shipping
FOB_destination
title_passage_rule_null

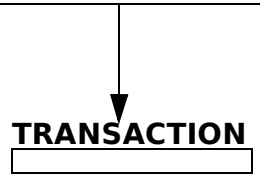
PURCHASE (2)

```
TRANSACTION *purchase_transaction =
    purchase_fixed_asset_transaction(
       full_name,
       street_address,
       arrived_date_time,
       purchase_invoice_amount,
       sales_tax,
       freight_in,
       fixed_asset_purchase_list(),
       account_payable() );

/* Returns true transaction_date_time */
char *purchase_transaction_refresh(
    purchase_transaction->
       transaction_amount,
    purchase_transaction->
       journal_list,
    purchase_transaction->memo,
    full_name,
    street_address,
    purchase_transaction->
       transaction_date_time );

void purchase_update(
    fixed_asset_purchase_total,
    purchase_invoice_amount,
    vendor_payment_total,
    purchase_amount_due,
       purchase_transaction->
          transaction_date_time,
       full_name,
       street_address,
       purchase_date_time );

LIST *inventory_purchase_list
LIST *specific_inventory_purchase_list
LIST *service_purchase_list
LIST *supply_purchase_list
LIST *prepaid_asset_purchase_list;
char *property_street_address;
```



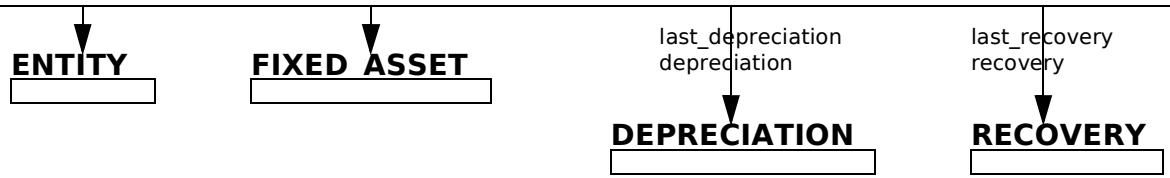
VENDOR_PAYMENT

```
/* Usage */
VENDOR_PAYMENT *vendor_payment_fetch(
    char *full_name,
    char *street_address,
    char *purchase_date_time,
    char *payment_date_time );

LIST *vendor_payment_list_fetch(
    full_name,
    street_address,
    purchase_date_time );

/* Input */
ENTITY *vendor_entity;
char *purchase_date_time;
char *payment_date_time;
double payment_amount;
int check_number;

/* Process */
VENDOR_PAYMENT *vendor_payment_new(
    full_name,
    street_address,
    purchase_date_time,
    payment_date_time );

char *vendor_payment_primary_where(
    full_name,
    street_address,
    purchase_date_time,
    payment_date_time );

char *vendor_payment_system_string(
    purchase_primary_where() );

LIST *vendor_payment_system_list(
    vendor_payment_system_string() );

VENDOR_PAYMENT *vendor_payment_parse(
    input );

double vendor_payment_total(
    LIST *vendor_payment_system_list() )
{
    return sum(
        A->
        vendor_payment->
        payment_amount );
}

TRANSACTION *vendor_payment_transaction(
    full_name,
    street_address,
    payment_date_time,
    payment_amount,
    check_number,
    account_payable(),
    account_cash(),
    account_uncleared_checks_string() );

LIST *vendor_payment_journal_list(
    payment_amount,
    check_number,
    account_payable(),
    account_cash(),
    account_uncleared_checks_string() );

void vendor_payment_insert(
    full_name,
    street_address,
    purchase_date_time,
    payment_date_time,
    payment_amount,
    check_number );

VENDOR_PAYMENT *vendor_payment_seek(
    vendor_payment_list,
    payment_date_time );

char *vendor_payment_update_system_string(
    void );
```



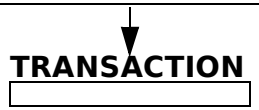
FIXED_ASSET_PURCHASE (1)

```
/* Usage */
LIST *fixed_asset_purchase_list_fetch(
    char *fixed_asset_purchase_depreciation_where(),
    boolean fetch_last_depreciation,
    boolean fetch_last_recovery );

/* Process */
char *fixed_asset_purchase_system_string(
    FIXED_ASSET_PURCHASE_TABLE,
    fixed_asset_purchase_depreciation_where,
    "service_placement_date" /* order */ );

LIST *fixed_asset_purchase_system_list(
    fixed_asset_purchase_system_string(),
    fetch_last_depreciation,
    fetch_last_recovery );

/* Usage */
LIST *fixed_asset_purchase_system_list(
    char *fixed_asset_purchase_system_string(),
    boolean fetch_last_depreciation,
    boolean fetch_last_recovery );

/* Process */
FILE *fixed_asset_purchase_input_pipe(
    char *fixed_asset_purchase_system_string() );

while (string_input(
       input[],
       fixed_asset_purchase_input_pipe() )
{
    list_set(
       list,
       fixed_asset_purchase_parse(
          input,
          fetch_last_depreciation,
          fetch_last_recovery ) );
}

pclose( fixed_asset_purchase_input_pipe() );

/* Usage */
FIXED_ASSET_PURCHASE *
    fixed_asset_purchase_fetch(
       char *asset_name,
       char *serial_label,
       boolean fetch_last_depreciation,
       boolean fetch_last_recovery );

/* Process */
char *fixed_asset_purchase_primary_where(
    asset_name,
    serial_label );

char *fixed_asset_purchase_system_string(
    FIXED_ASSET_PURCHASE_TABLE,
    fixed_asset_purchase_primary_where(),
    (char *)0 /* order */ );

char *string_pipe_fetch(
    fixed_asset_purchase_system_string() );

FIXED_ASSET_PURCHASE *
    fixed_asset_purchase_parse(
       string_pipe_fetch() /* input */,
       fetch_last_depreciation,
       fetch_last_recovery );

/* Usage */
FIXED_ASSET_PURCHASE *fixed_asset_purchase_parse(
    char *input,
    boolean fetch_last_depreciation,
    boolean fetch_last_recovery );

/* Process */
FIXED_ASSET_PURCHASE *fixed_asset_purchase_new(
    asset_name,
    serial_number );

enum depreciation_method depreciation_method =
    depreciation_method_resolve();

if ( fetch_last_depreciation )
{
    last_deprecation =
       depreciation_fetch(
          asset_name,
          serial_label,
          depreciation_prior_depreciation_date()
             /* depreciation_date */ );
}

if ( fetch_last_recovery )
{
    last_recovery =
       recovery_fetch(
          asset_name,
          serial_label,
          recovery_prior_tax_year()
             /* tax_year */ );
}

/* Attributes */
char *asset_name;
char *serial_label;
ENTITY *vendor_entity;
char *purchase_date_time;
char *service_placement_date;
double fixed_asset_cost;
int units_produced_so_far;
char *disposal_date;
char *recovery_class_year_string;
char *recovery_method;
char *recovery_convention;
char *recovery_system;
enum depreciation_method depreciation_method;
int estimated_useful_life_years;
int estimated_useful_life_units;
int estimated_residual_value;
int declining_balance_n;
double cost_basis;
double finance_accumulated_depreciation;
double tax_adjusted_basis;
FIXED_ASSET *fixed_asset;
DEPRECIATION *last_depreciation;
DEPRECIATION *depreciation;
RECOVERY *last_recovery;
RECOVERY *recovery;
TRANSACTION *depreciation_transaction;
```



FIXED_ASSET_PURCHASE (2)

```
/* Usage */
FIXED_ASSET_PURCHASE *fixed_asset_purchase_new(
    char *asset_name,
    char *serial_number );

/* Process */
FIXED_ASSET_PURCHASE *fixed_asset_purchase_calloc(
   void );

FIXED_ASSET *fixed_asset =
    fixed_asset_fetch(
       asset_name );

/* Usage */
char *fixed_asset_purchase_primary_where(
   char *asset_name,
   char *serial_label );

/* Process */
char *fixed_asset_name_escape( asset_name );

/* Usage */
/* Returns fixed_asset_purchase_list_fetch() */
LIST *fixed_asset_purchase_depreciation_list(
   LIST *fixed_asset_purchase_list_fetch(),
   char *fixed_asset_purchase_depreciation_date() );

/* Process */
ENTITY_SELF *entity_self =
   entity_self_fetch(
      0 /* not fetch_entity_boolean */ );

char *account_depreciation_string(
   ACCOUNT_DEPRECIATION_KEY,
   __FUNCTION__ );

char *account_accumulated_depreciation_string(
   ACCOUNT_ACCUMULATED_DEPRECIATION_KEY,
   __FUNCTION__ );

for FIXED_ASSET_PURCHASE *fixed_asset_purchase in
   fixed_asset_purchase_list_fetch()
{
   char *prior_depreciation_date =
      fixed_asset_purchase_prior_depreciation_date(
         DEPRECIATION *
            fixed_asset_purchase->
            last_depreciation );

   fixed_asset_purchase->depreciation =
      DEPRECIATION *depreciation_evaluate(
         asset_name,
         serial_label,
         depreciation_method_evaluate(),
         service_placement_date,
         prior_depreciation_date,
         cost_basis,
         units_produced_so_far,
         estimated_residual_value,
         estimated_useful_life_years,
         estimated_useful_life_units,
         declining_balance_n,
         finance_accumulated_depreciation
            /* prior_accumulated_depreciation */ );

   if (  fixed_asset_purchase->
            depreciation
   &&  fixed_asset_purchase->
            depreciation->
            depreciation_amount > 0.0 )
   {
      TRANSACTION *depreciation_transaction(
         entity_self->full_name,
         entity_self->street_address,
         fixed_asset_purchase_depreciation_date(),
         depreciation->depreciation_amount,
         account_depreciation_string(),
         account_accumulated_depreciation_string() );
   }
}

/* Usage */
void fixed_asset_purchase_list_update(
   LIST *fixed_asset_purchase_list );

/* Process */
for FIXED_ASSET_PURCHASE *fixed_asset_purchase in
   fixed_asset_purchase_list
{
   void fixed_asset_purchase_update(
      fixed_asset_purchase );
}

/* Usage */
void fixed_asset_purchase_update(
   FIXED_ASSET_PURCHASE *fixed_asset_purchase );

/* Process */
FILE *fixed_asset_purchase_update_pipe(
   char *FIXED_ASSET_PURCHASE_TABLE,
   char *FIXED_ASSET_PURCHASE_PRIMARY_KEY );

void fixed_asset_purchase_update_execute(
   FILE *fixed_asset_purchase_update_pipe(),
   double fixed_asset_purchase->cost_basis,
   double fixed_asset_purchase->
      finance_accumulated_depreciation,
   double fixed_asset_purchase->tax_adjusted_basis,
   char *fixed_asset_purchase->asset_name,
   char *fixed_asset_purchase->serial_label );

void pclose( fixed_asset_purchase_update_pipe() );
```



FIXED_ASSET_PURCHASE (3)

```
/* Usage */
LIST *fixed_asset_purchase_transaction_list_extract(
    LIST *fixed_asset_purchase_depreciation_list() );

/* Process */
for FIXED_ASSET_PURCHASE *fixed_asset_purchase in
    fixed_asset_purchase_depreciation_list
{
    if ( fixed_asset_purchase->depreciation_transaction )
    {
        list_set(
            transaction_list,
            fixed_asset_purchase->
                depreciation_transaction );
    }
}

/* Usage */
LIST *fixed_asset_purchase_depreciation_list_extract(
    LIST *fixed_asset_purchase__depreciation_list );

/* Process */
for FIXED_ASSET_PURCHASE *fixed_asset_purchase in
    fixed_asset_purchase_depreciation_list
{
    if (fixed_asset_purchase->depreciation
    &&fixed_asset_purchase->
          depreciation_transaction )
    {
        fixed_asset_purchase->
            depreciation->
            transaction_date_time =
                fixed_asset_purchase->
                    depreciation_transaction->
                    transaction_date_time;

        list_set(
            depreciation_list,
            fixed_asset_purchase->depreciation );
    }
}

/* Usage */
void fixed_asset_purchase_transaction_list_insert(
    /* May reset depreciation_transaction->
       transaction_date_time */
    LIST *
       fixed_asset_purchase_transaction_list_extract() );

/* Process */
void transaction_list_insert(
    fixed_asset_purchase_transaction_list_extract,
    1 /* insert_journal_list_boolean  */,
    1 /* transaction_lock_boolean */ );

/* Usage */
void fixed_asset_purchase_depreciation_list_insert(
    LIST *
    fixed_asset_purchase_depreciation_list_extract() );

/* Process */
void depreciation_list_insert(
    fixed_asset_purchase_depreciation_list_extract );

/* Driver */
void fixed_asset_purchase_depreciation_display(
    LIST *fixed_asset_purchase_depreciation_list() );

/* Driver */
void fixed_asset_purchase_depreciation_insert(
    LIST *fixed_asset_purchase_depreciation_list() );

/* Process */
LIST *fixed_asset_purchase_transaction_list_extract(
    LIST *fixed_asset_purchase_depreciation_list );

/* May reset transaction->transaction_date_time */
void fixed_asset_purchase_transaction_list_insert(
    fixed_asset_purchase_transaction_list_extract() );

LIST *fixed_asset_purchase_depreciation_list_extract(
    LIST *fixed_asset_purchase_depreciation_list );

void fixed_asset_purchase_depreciation_list_insert(
    fixed_asset_purchase_depreciation_list_extract() );

void transaction_list_html_display(
    fixed_asset_purchase_transaction_list_extract() );

/* Public */
char *fixed_asset_purchase_depreciation_where(
    char *fixed_asset_purchase_depreciation_date() );

char *fixed_asset_purchase_depreciation_date(
    void );

double fixed_asset_purchase_cost_basis(
    double fixed_asset_cost );

double fixed_asset_purchase_tax_adjusted_basis(
    double fixed_asset_cost );

char *fixed_asset_purchase_system_string(
    char *FIXED_ASSET_PURCHASE_TABLE,
    char *where,
    char *order );

double tax_accumulated_depreciation =
    recovery_accumulated();

void fixed_asset_purchase_finance_fetch_update(
    char *asset_name,
    char *serial_label );

RECOVERY *recovery =
    recovery_evaluate();

void fixed_asset_purchase_cost_fetch_update(
    char *asset_name,
    char *serial_label );

LIST *fixed_asset_purchase_list_cost_recover(
    LIST *fixed_asset_purchase_list,
    int tax_year );

LIST *fixed_asset_purchase_cost_recovery_list(
    LIST *fixed_asset_purchase_list );

void
fixed_asset_purchase_negate_depreciation_amount(
    LIST *fixed_asset_purchase_list );

void
fixed_asset_purchase_negate_recovery_amount(
    LIST *fixed_asset_purchase_list );
```

FIXED_ASSET

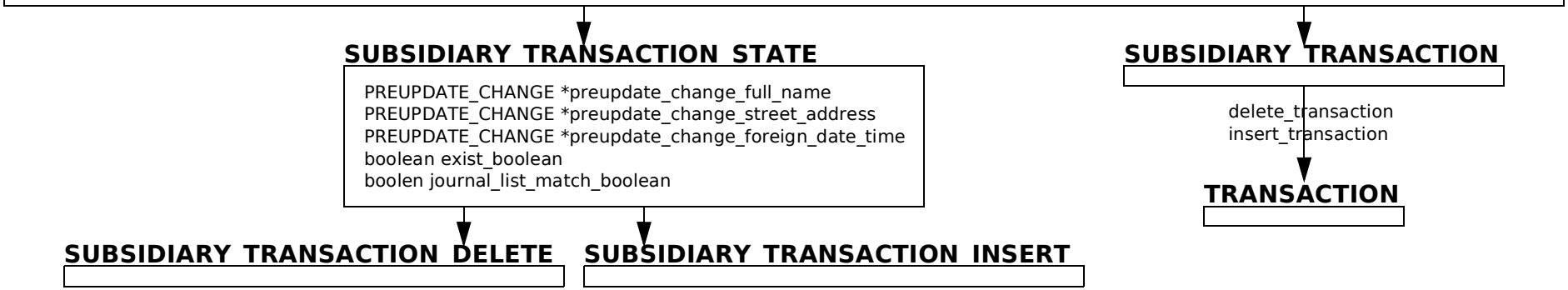
```
/* Usage */
FIXED_ASSET *fixed_asset_fetch(
    char *asset_name );

/* Process */
char *fixed_asset_primary_where(
    asset_name );

char *fixed_asset_system_string(
    char *FIXED_ASSET_TABLE,
    char *fixed_asset_primary_where() );

char *string_pipe_input(
    fixed_asset_system_string() );

FIXED_ASSET *fixed_asset_parse(
    string_pipe_input() );

/* Usage */
FIXED_ASSET *fixed_asset_parse(
    char *input );

/* Process */
FIXED_ASSET *fixed_asset_new(
    strdup( asset_name );

/* Usage */
FIXED_ASSET *fixed_asset_new(
    char *asset_name );

/* Process */
FIXED_ASSET *fixed_asset_calloc(
    void );

/* Usage */
char *fixed_asset_primary_where(
    char *asset_name );

/* Process */
char *fixed_asset_name_escape(
    char *asset_name );

/* Attributes */
char *asset_name;
char *account_name;
char *cost_recovery_period_string;
char *cost_recovery_method;
char *cost_recovery_conversion;
double activity_energy_kilowatt_draw;
double activity_depreciation_per_hour;
```

PRIOR_FIXED_ASSET (1)

```
/* Usage */
PRIOR_FIXED_ASSET *
    prior_fixed_asset_fetch(
        char *asset_name,
        char *state,
        char *preupdate_full_name,
        char *preupdate_street_address,
        char *preupdate_purchase_date_time );

/* Process */
char *prior_fixed_asset_primary_where(
    const char *PRIOR_FIXED_ASSET_PRIMARY_KEY,
    char *asset_name );

char *appaserver_system_string(
    PRIOR_FIXED_ASSET_SELECT,
    PRIOR_FIXED_ASSET_TABLE,
    prior_fixed_asset_primary_where() );

char *string_pipe_fetch( appaserver_system_string() );

PRIOR_FIXED_ASSET *
    prior_fixed_asset_parse(
        asset_name,
        string_pipe_fetch() /* input */ );

if (    prior_fixed_asset_parse()->fixed_asset_cost
&&  prior_fixed_asset_parse()->asset_account )
{
    ACCOUNT *debit_account =
        account_fetch(
            prior_fixed_asset_parse()->
                asset_account,
            1 /* fetch_subclassification */,
            1 /* fetch_element */ );

    char *account_equity_string(
        ACCOUNT_EQUITY_KEY,
        __FUNCTION__ );

    ACCOUNT *credit_account =
        account_fetch(
            account_equity_string(),
            1 /* fetch_subclassification */,
            1 /* fetch_element */ );

    LIST *journal_binary_list(
        (char *)0 /* full_name */,
        (char *)0 /* street_address */,
        (char *)0 /* transaction_date_time */,
        prior_fixed_asset_parse()->fixed_asset_cost
            /* transaction_amount */,
        debit_account,
        credit_account );
}

SUBSIDIARY_TRANSACTION_STATE *
    subsidiary_transaction_state_new(
        "preupdate_full_name"
            /* preupdate_full_name_placeholder */,
        "preupdate_street_address"
            /* preupdate_street_address_placeholder */,
        "preupdate_purchase_date_time"
            /* preupdate_foreign_date_time_placeholder */,
        state,
        preupdate_full_name,
        preupdate_street_address,
        preupdate_purchase_date_time
            /* preupdate_foreign_date_time */,
        prior_fixed_asset_parse()->
            full_name,
        prior_fixed_asset_parse()->
            street_address,
        prior_fixed_asset_parse()->
            purchase_date_time
            /* foreign_date_time */,
        journal_binary_list()
            /* insert_journal_list */ );

SUBSIDIARY_TRANSACTION *
    subsidiary_transaction_new(
        PRIOR_FIXED_ASSET_TABLE
            /* foreign_table_name */,
        "full_name"
            /* foreign_full_name_column */,
        "street_address"
            /* foreign_street_address_column */,
        "purchase_date_time"
            /* foreign_date_time_column */,
        prior_fixed_asset_parse()->purchase_date_time
            /* prior_transaction_date_time */,
        journal_binary_list()
            /* insert_journal_list */,
        prior_fixed_asset_parse()->fixed_asset_cost
            /* foreign_amount */,
        asset_name /* transaction_memo */,
        subsidiary_transaction_state_new()->
            subsidary_transaction_insert,
        subsidiary_transaction_state_new()->
            subsidiary_transaction_delete );
```

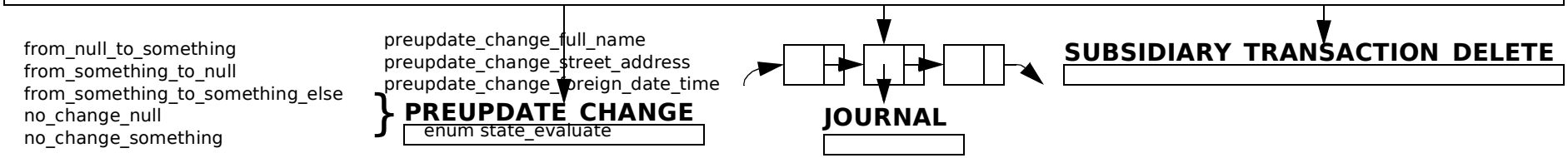
PRIOR_FIXED_ASSET (2)

```
/* Usage */
PRIOR_FIXED_ASSET *prior_fixed_asset_parse(
    char *asset_name,
    char *input );

/* Process */
PRIOR_FIXED_ASSET *prior_fixed_asset_new(
    asset_name );

/* Usage */
PRIOR_FIXED_ASSET *prior_fixed_asset_new(
    char *asset_name );

/* Process */
PRIOR_FIXED_ASSET *prior_fixed_asset_calloc(
    void );

/* Driver */
void subsidiary_transaction_execute(
    application_name,
    prior_fixed_asset->
      subsidiary_transaction->
        delete_transaction,
    prior_fixed_asset->
      subsidiary_transaction->
        insert_transaction,
    prior_fixed_asset->
      subsidiary_transaction->
      update_template );

/* Attributes */
char *asset_name;
char *full_name;
char *street_address;
char *purchase_date_time;
double fixed_asset_cost;
char *asset_account;
ACCOUNT *debit_account;
char *account_equity_string;
ACCOUNT *credit_account;
LIST *journal_binary_list;
SUBSIDIARY_TRANSACTION_STATE *
    subsidiary_transaction_state;
SUBSIDIARY_TRANSACTION *
    subsidiary_transaction;
```



SUBSIDIARY_TRANSACTION_STATE

```
/* Usage */
SUBSIDIARY_TRANSACTION_STATE *
   subsidiary_transaction_state_new(
      const char *
         preupdate_full_name_placeholder,
      const char *
         preupdate_street_address_placeholder,
      const char *
         preupdate_foreign_date_time_placeholder,
      char *state,
      char *preupdate_full_name,
      char *preupdate_street_address,
      char *preupdate_foreign_date_time,
      char *full_name,
      char *street_address,
      char *foreign_date_time,
      LIST *insert_journal_list );

/* Process */
SUBSIDIARY_TRANSACTION_STATE *
   subsidiary_transaction_state_calloc(
      void );

PREUPDATE_CHANGE *
   preupdate_change_full_name =
      preupdate_change_new(
         APPASERVER_INSERT_STATE,
         APPASERVER_PREDELETE_STATE,
         state,
         preupdate_full_name
            /* preupdate_datum */,
         full_name
            /* postupdate_datum */,
         preupdate_full_name_placeholder,
            /* preupdate_placeholder_name */ );

PREUPDATE_CHANGE *
   preupdate_change_street_address =
      preupdate_change_new(
         APPASERVER_INSERT_STATE,
         APPASERVER_PREDELETE_STATE,
         state,
         preupdate_street_address,
         street_address,
         preupdate_street_address_placeholder );

PREUPDATE_CHANGE *
   preupdate_change_foreign_date_time =
      preupdate_change_new(
         APPASERVER_INSERT_STATE,
         APPASERVER_PREDELETE_STATE,
         state,
         preupdate_foreign_date_time,
         foreign_date_time,
         preupdate_foreign_date_time_placeholder );

LIST *old_journal_list =
   journal_transaction_list(
      JOURNAL_SELECT,
      JOURNAL_TABLE,
      full_name,
      street_address,
      foreign_date_time
         /* transaction_date_time */ );

boolean subsidiary_transaction_state_exist_boolean(
   LIST *old_journal_list );

if ( subsidiary_transaction_state_exist_boolean() )
{
   boolean journal_list_match_boolean(
      insert_journal_list /* journal1_list */,
      old_journal_list /* journal2_list */ );
}

SUBSIDIARY_TRANSACTION_INSERT *
   subsidiary_transaction_insert_new(
      preupdate_change_full_name,
      preupdate_change_street_address,
      preupdate_change_foreign_date_time,
      journal_list_match_boolean() );

SUBSIDIARY_TRANSACTION_DELETE *
   subsidiary_transaction_delete_new(
      preupdate_change_full_name,
      preupdate_change_street_address,
      preupdate_change_foreign_date_time,
      journal_list_match_boolean() );

/* Attributes */
PREUPDATE_CHANGE *
   preupdate_change_full_name;
PREUPDATE_CHANGE *
   preupdate_change_street_address;
PREUPDATE_CHANGE *
   preupdate_change_foreign_date_time;
LIST *old_journal_list;
boolean exist_boolean;
boolean journal_list_match_boolean;
SUBSIDIARY_TRANSACTION_INSERT *
   subsidiary_transaction_insert;
SUBSIDIARY_TRANSACTION_DELETE *
   subsidiary_transaction_delete;
```



SUBSIDIARY_TRANSACTION_INSERT

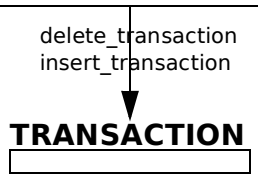
```
/* Usage */
SUBSIDIARY_TRANSACTION_INSERT *
    subsidiary_transaction_insert_new(
        PREUPDATE_CHANGE *
            preupdate_change_full_name,
        PREUPDATE_CHANGE *
            preupdate_change_street_address,
        PREUPDATE_CHANGE *
            preupdate_change_foreign_date_time,
        boolean journal_list_match_boolean );

/* Process */
if (    preupdate_change_full_name->
            no_change_boolean
&&  preupdate_change_street_address->
            no_change_boolean
&&  preupdate_change_foreign_date_time->
            no_change_boolean
&&  journal_list_match_boolean )
{
    return null;
}

SUBSIDIARY_TRANSACTION_INSERT *
    subsidiary_transaction_insert_calloc(
        void );

full_name =
    preupdate_change_full_name->
        new_datum );

if ( !full_name ) return null;

street_address =
    preupdate_change_street_address->
        new_datum );

if ( !street_address ) return null;

transaction_date_time =
    preupdate_change_foreign_date_time->
        new_datum;

if ( !transaction_date_time ) return null;

/* Attributes */
char *full_name;
char *street_address;
char *transaction_date_time;
```

SUBSIDIARY_TRANSACTION_DELETE

```
/* Usage */
SUBSIDIARY_TRANSACTION_DELETE *
    subsidiary_transaction_delete_new(
        PREUPDATE_CHANGE *
            preupdate_change_full_name,
        PREUPDATE_CHANGE *
            preupdate_change_street_address,
        PREUPDATE_CHANGE *
            preupdate_change_foreign_date_time,
        boolean journal_list_match_boolean );

/* Process */
if (    preupdate_change_full_name->
            no_change_boolean
&&  preupdate_change_street_address->
            no_change_boolean
&&  preupdate_change_foreign_date_time->
            no_change_boolean
&&  journal_list_match_boolean )
{
    return null;
}

SUBSIDIARY_TRANSACTION_DELETE *
    subsidiary_transaction_delete_calloc(
        void );

full_name =
    preupdate_change_full_name->
        prior_datum;

if ( !full_name ) return null;

street_address =
    preupdate_change_street_address->
        prior_datum;

if ( !street_address ) return null;

transaction_date_time =
    preupdate_change_foreign_date_time->
        prior_datum;

if ( !transaction_date_time ) return null;

/* Attributes */
char *full_name;
char *street_address;
char *transaction_date_time;
```

SUBSIDIARY_TRANSACTION (1)

```
/* Usage */
SUBSIDIARY_TRANSACTION *
    subsidiary_transaction_new(
        const char *foreign_table_name,
        const char *foreign_full_name_column,
        const char *foreign_street_address_column,
        const char *foreign_date_time_column,
        char *prior_transaction_date_time,
        LIST *insert_journal_list,
        double foreign_amount,
        char *transaction_memo,
        SUBSIDIARY_TRANSACTION_INSERT *
            subsidiary_transaction_state->
                subsidiary_transaction_insert,
        SUBSIDIARY_TRANSACTION_DELETE *
            subsidiary_transaction_state->
                subsidiary_transaction_delete );

/* Process */
SUBSIDIARY_TRANSACTION *
    subsidiary_transaction_calloc(
        void );

if ( subsidiary_transaction_delete )
{
    TRANSACTION *delete_transaction =
        transaction_new(
            subsidiary_transact_delete->
                full_name,
            subsidiary_transaction_delete->
                street_address,
            subsidiary_transaction_delete->
                transaction_date_time );
}

if (    foreign_amount > 0.0
&&  subsidiary_transaction_insert )
{
    TRANSACTION *insert_transaction =
        transaction_new(
            subsidiary_transact_insert->
                full_name,
            subsidiary_transaction_insert->
                street_address,
            subsidiary_transaction_insert->
                transaction_date_time );

    insert_transaction->journal_list = insert_journal_list;
    insert_transaction->transaction_amount =
        foreign_amount;

    insert_transaction->memo = transaction_memo;

    char *subsidiary_transaction_update_template(
        foreign_table_name,
        foreign_full_name_column,
        foreign_street_address_column,
        foreign_date_time_column,
        subsidiary_transaction_insert->full_name,
        subsidiary_transaction_insert->street_address,
        subsidiary_transaction_insert->
            transaction_date_time
                /* foreign_date_time */,
        prior_transaction_date_time );
}

/* Attributes */
TRANSACTION *delete_transaction;
TRANSACTION *insert_transaction;
char *update_template;
```



SUBSIDIARY_TRANSACTION (2)

```
/* Usage */
char *subsidiary_transaction_update_template(
    const char *foreign_table_name,
    const char *foreign_full_name_column,
    const char *foreign_street_address_column,
    const char *foreign_date_time_column,
    char *full_name,
    char *street_address,
    char *foreign_date_time,
    char *prior_transaction_date_time );

/* Process */
if ( string_strcmp(
        foreign_date_time,
        prior_transaction_date_time ) == 0 )
{
    return null;
}

snprintf(
    update_template[],
    sizeof ( update_template ),
    “update %s “
    “set %s = ‘%cs’ “
    “where %s = ‘%s’ and “
    “%s = ‘%s’ and “
    “%s = ‘%s’;”,
foreign_table_name,
foreign_date_time_column,
‘%’,
foreign_full_name_column,
full_name,
foreign_street_address_column,
street_address,
foreign_date_time_column,
foreign_date_time );
```

SUBSIDIARY_TRANSACTION (3)

```
/* Driver */
void subsidiary_transaction_execute(
    char *application_name,
    TRANSACTION *
       subsidiary_transaction->
          delete_transaction,
    TRANSACTION *
       subsidiary_transaction->
          insert_transaction,
    char *subsidiary_transaction->update_template );

/* Process */
if ( delete_transaction )
{
    void transaction_delete(
       delete_transaction->full_name,
       delete_transaction->street_address,
       delete_transaction->transaction_date_time );
}

if ( insert_transaction )
{
    char *transaction_date_time =
       transaction_insert(
          insert_transaction->full_name,
          insert_transaction->street_address,
          insert_transaction->transaction_date_time,
          insert_transaction->transaction_amount,
          0 /* check_number */,
          insert_transaction->memo,
          'n' /* lock_transaction_yn not present */,
          insert_transaction->journal_list,
          1 /* insert_journal_list_boolean */ );

    int strcmp(
       insert_transaction->transaction_date_time,
       transaction_date_time );

    if ( strcmp() != 0 )
    {
       snprintf(
          update_statement[],
          sizeof ( update_statement ),
          update_template,
          transaction_date_time );

       void update_statement_execute(
          SQL_EXECUTABLE,
          application_name,
          update_statement );
    }
}
```

RECOVERY (1)

```
/* Usage */
RECOVERY *recovery_evaluate(
    char *asset_name,
    char *serial_label,
    int tax_year,
    double fixed_asset_purchase->
       cost_basis,
    char *fixed_asset_purchase->
       service_placement_date
          /* service_placement_date_string */,
    char *fixed_asset_purchase->
       disposal_date
          /* disposal_date_string */,
    char *fixed_asset_purchase->
       cost_recovery_class_year_string,
    char *fixed_asset_purchase->
       cost_recovery_method,
    char *fixed_asset_purchase->
       cost_recovery_convention,
    char *fixed_asset_purchase->
       cost_recovery_system );

/* Process */
RECOVERY *recovery_new(
    asset_name,
    serial_label,
    tax_year );

DATE *recovery_service_placement_date(
    char *service_placement_date_string );

int service_placement_month =
    date_month(
       recovery_service_placement_date() );

int service_placement_year =
    date_year(
       recovery_service_placement_date() );

if ( disposal_date_string && *disposal_date_string )
{
    DATE *recovery_disposal_date(
       disposal_date_string );

    int disposal_month =
       date_month(
          recovery_disposal_date() );

    int disposal_year =
       date_year(
          recovery_disposal_date() );
}

int recovery_year(
    tax_year,
    service_placement_year,
    disposal_year );

if ( !recovery_year() ) return this;

double recovery_class_year(
    char *cost_recovery_class_year_string );

if ( string_strcmp(
       cost_recovery_method,
       RECOVERY_METHOD_STRAIGHT_LINE ) == 0 )
{
    RECOVERY_STRAIGHT_LINE *recovery_straight_line =
       recovery_straight_line_new(
          cost_basis,
          service_placement_month,
          disposal_month,
          cost_recovery_convention,
          cost_recovery_system,
          recovery_year(),
          recovery_class_year() );
}
else
if ( string_strcmp(
       cost_recovery_method,
       RECOVERY_METHOD_ACCELERATED ) ==  0 )
{
    RECOVERY_ACCELERATED *recovery_accelerated =
       recovery_accelerated_new(
          tax_year,
          cost_basis,
          service_placement_month,
          recovery_period_years(),
          cost_recovery_convention,
          cost_recovery_system,
          recovery_year(),
          recovery_class_year() );
}

double recovery_rate(
    RECOVERY_STRAIGHT_LINE *
       recovery_straight_line,
    RECOVERY_ACCELERATED *
       recovery_accelerated );

double recovery_amount(
    RECOVERY_STRAIGHT_LINE *
       recovery_straight_line,
    RECOVERY_ACCELERATED *
       recovery_accelerated );

/* Attributes */
char *asset_name;
char *serial_label;
int tax_year;
double cost_basis;
DATE *service_placement_date;
int service_placement_month;
int service_placement_year;
DATE *disposal_date;
int disposal_month;
int disposal_year;
int year;
double class_year;
RECOVERY_STRAIGHT_LINE *recovery_straight_line;
RECOVERY_ACCELERATED *recovery_accelerated;
double rate;
double amount;
```

service_placement_date
disposal_date

DATE

RECOVERY_STRAIGHT_LINE

RECOVERY_ACCELERATED

RECOVERY (2)

```
/* Usage */
RECOVERY *recovery_fetch(
    char *asset_name,
    char *serial_label,
    int tax_year );

/* Process */
char *recovery_primary_where(
    char *asset_name,
    char *serial_label,
    int tax_year );

char *appaserver_system_striing(
    RECOVERY_SELECT,
    RECOVERY_TABLE,
    recovery_primary_where() );

char *string_pipe_input(
    appaserver_system_string() );

RECOVERY *recovery_parse(
    string_pipe_input() );

/* Usage */
LIST *recovery_fetch_list(
    char *asset_name,
    char *serial_label );

/* Process */
char *fixed_asset_purchase_primary_where(
    asset_name,
    serial_label );

char *appaserver_system_striing(
    RECOVERY_SELECT,
    RECOVERY_TABLE,
    fixed_asset_purchase_primary_where() );

LIST *recovery_system_list(
    appaserver_system_string() );

/* Usage */
LIST *recovery_system_list(
    char *appaserver_system_string() );

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

FILE *appaserver_input_pipe(
    appaserver_system_string );

while ( string_input( input[], appaserver_input_pipe() ) )
{
    list_set(
       list,
       recovery_parse( input ) );
}
pclose( appaserver_input_pipe() );

/* Usage */
RECOVERY *recovery_parse(
    char *input );

/* Process */
RECOVERY *recovery_new(
    strdup( asset_name ),
    strdup( serial_label ),
    atoi( tax_year ) );

/* Usage */
RECOVERY *recovery_new(
    char *asset_name,
    char *serial_label,
    int tax_year );

/* Process */
RECOVERY *recovery_calloc(
    void );

/* Usage */
int recovery_year(
    tax_year,
    service_placement_year,
    disposal_year );

/* Process */
if ( disposal_year )
{
    if ( tax_year > disposal_year )
    {
       year = 0;
    }
    else
    {
       year =
          tax_year -
          disposal_year +
          1;
    }
}
else
{
    year =
       tax_year -
       service_placement_year +
       1;
}
```

RECOVERY (3)

```
/* Usage */
void recovery_list_insert(
    char *RECOVERY_TABLE,
    LIST *recovery_list );

/* Process */
void recovery_insert_open(
    char *RECOVERY_TABLE );

void recovery_insert(
    FILE *recovery_insert_open(),
    char *asset_name,
    char *serial_label,
    int tax_year,
    double recovery_rate(),
    double recovery_amount() );

/* Usage */
void recovery_update(
    char *asset_name,
    char *serial_label,
    int tax_year,
    double recovery_rate(),
    double recovery_amount() );

/* Process */
FILE *recovery_update_open(
    char *RECOVERY_TABLE );

char *fixed_asset_name_escape(
    asset_name );

/* Usage */
FILE *recovery_delete_open(
    char *RECOVERY_TABLE );

/* Process */

/* Public */
char *recovery_subquery_where(
    int tax_year );

void recovery_fixed_asset_list_set(
    LIST *fixed_asset_list,
    int tax_year );

void recovery_fixed_asset_list_display(
    FILE *output_pipe,
    LIST *fixed_asset_list );

int recovery_prior_tax_year(
    char *RECOVERY_TABLE );
```

RECOVERY_STRAIGHT_LINE

```
/* Usage */
RECOVERY_STRAIGHT_LINE *
    recovery_straight_line_new(
       double cost_basis,
       int service_placement_month,
       int disposal_month,
       char *cost_recovery_convention,
       char *cost_recovery_system,
       int recovery_year(),
       double recovery_class_year() );

/* Process */
RECOVERY_STRAIGHT_LINE *
    recovery_straight_line_calloc(
       void );

if ( string_strcmp(
       cost_recovery_convention,
       RECOVERY_CONVENTION_HALF_YEAR ) == 0 )
{
    double rate =
       double recovery_straight_line_half_year_rate(
          char *cost_recovery_system,
          int recovery_year,
          double recovery_class_year );
}
else
if ( string_strcmp(
       cost_recovery_convention,
       RECOVERY_CONVENTION_MID_MONTH ) == 0 )
{
    double rate =
       double recovery_straight_line_mid_month_rate(
          int service_placement_month,
          int disposal_month,
          char *cost_recovery_system,
          int recovery_year,
          double recovery_class_year );
}
else
if ( string_strcmp(
       cost_recovery_convention,
       RECOVERY_CONVENTION_MID_QUARTER ) == 0 )
{
    double rate =
       double recovery_straight_line_mid_quarter_rate(
          int service_placement_month,
          int disposal_month,
          char *cost_recovery_system,
          int recovery_year,
          double recovery_class_year );
}

double recovery_straight_line_amount(
    double cost_basis,
    double rate );

/* Attributes */
double rate;
double amount;
```

RECOVERY_ACCELERATED

```
/* Usage */
RECOVERY_ACCELERATED_LINE *
    recovery_accelerated_new(
       double cost_basis,
       int service_placement_month,
       int disposal_month,
       char *cost_recovery_convention,
       char *cost_recovery_system,
       int recovery_year(),
       double recovery_class_year() );

/* Process */
RECOVERY_ACCELERATED *
    recovery_accelerated_calloc(
       void );

if ( string_strcmp(
       cost_recovery_convention,
       RECOVERY_CONVENTION_HALF_YEAR ) == 0 )
{
    double rate =
       double recovery_accelerated_half_year_rate(
          char *cost_recovery_system,
          int recovery_year,
          double recovery_class_year );
}
else
if ( string_strcmp(
       cost_recovery_convention,
       RECOVERY_CONVENTION_MID_MONTH ) == 0 )
{
    double rate =
       double recovery_accelerated_mid_month_rate(
          int service_placement_month,
          int disposal_month,
          char *cost_recovery_system,
          int recovery_year,
          double recovery_class_year );
}
else
if ( string_strcmp(
       cost_recovery_convention,
       RECOVERY_CONVENTION_MID_QUARTER ) == 0 )
{
    double rate =
       double recovery_accelerated_mid_quarter_rate(
          int service_placement_month,
          int disposal_month,
          char *cost_recovery_system,
          int recovery_year,
          double recovery_class_year );
}

double recovery_accelerated_amount(
    double cost_basis,
    double rate );

/* Attributes */
double rate;
double amount;
```

DEPRECIATION (1)

```
/* Usage */
DEPRECIATION *depreciation_evaluate(
    char *asset_name,
    char *serial_label,
    enum depreciation_method
      depreciation_method_evaluate(),
    char *service_placement_date,
    char *depreciation_prior_depreciation_date(),
    char *fixed_asset_purchase_depreciation_date()
      /* depreciation_date */,
    double cost_basis,
    int units_produced_so_far,
    int estimated_residual_value,
    int estimated_useful_life_years,
    int estimated_useful_life_units,
    int declining_balance_n,
    double fixed_asset_purchase->
      finance_accumulated_depreciation
        /* prior_accumulated_depreciation */ );

/* Process */
DEPRECIATION *depreciation_new(
    asset_name,
    serial_label,
    depreciation_date );

int depreciation_units_produced(
    asset_name,
    serial_label,
    depreciation_method,
    units_produced_so_far );

double depreciation_amount(
    depreciation_method,
    service_placement_date,
    depreciation_prior_depreciation_date,
    depreciation_date,
    cost_basis,
    depreciation_units_produced(),
    estimated_residual_value,
    estimated_useful_life_years,
    estimated_useful_life_units,
    declining_balance_n,
    prior_accumulated_depreciation );

if ( depreciation_amount() <= 0.0 ) return NULL;

double finance_accumulated_depreciation =
   double depreciation_accumulated(
      double prior_accumulated_depreciation,
      double depreciation_amount() );

static ENTITY_SELF *entity_self =
    entity_self_fetch(
       0 /* not fetch_entity_boolean */ );

/* Usage */
DEPRECIATION *depreciation_parse(
    char *input );

/* Process */
DEPRECIATION *depreciation_new(
    asset_name,
    serial_label,
    depreciation_date );

if ( *full_name[] )
{
    ENTITY_SELF *entity_self =
      ENTITY_SELF *entity_self_new(
        strdup( full_name ),
        strdup( street_address ) );
}

/* Usage */
DEPRECIATION *depreciation_new(
    char *asset_name,
    char *serial_label,
    char *depreciation_date );

/* Process */
DEPRECIATION *depreciation_calloc(
    void );

/* Attributes */
char *asset_name;
char *serial_label;
char *depreciation_date;
enum depreciation_method depreciation_method;
int units_produced;
double amount;
double finance_accumulated_depreciation;
ENTITY_SELF *entity_self;
char *transaction_date_time;
```

enum depreciation_method

straight_line,
double_declining_balance,
n_declining_balance,
sum_of_years_digits,
units_of_production,
not_depreciated

ENTITY_SELF

char *full_name
char *street_address

DEPRECIATION (2)

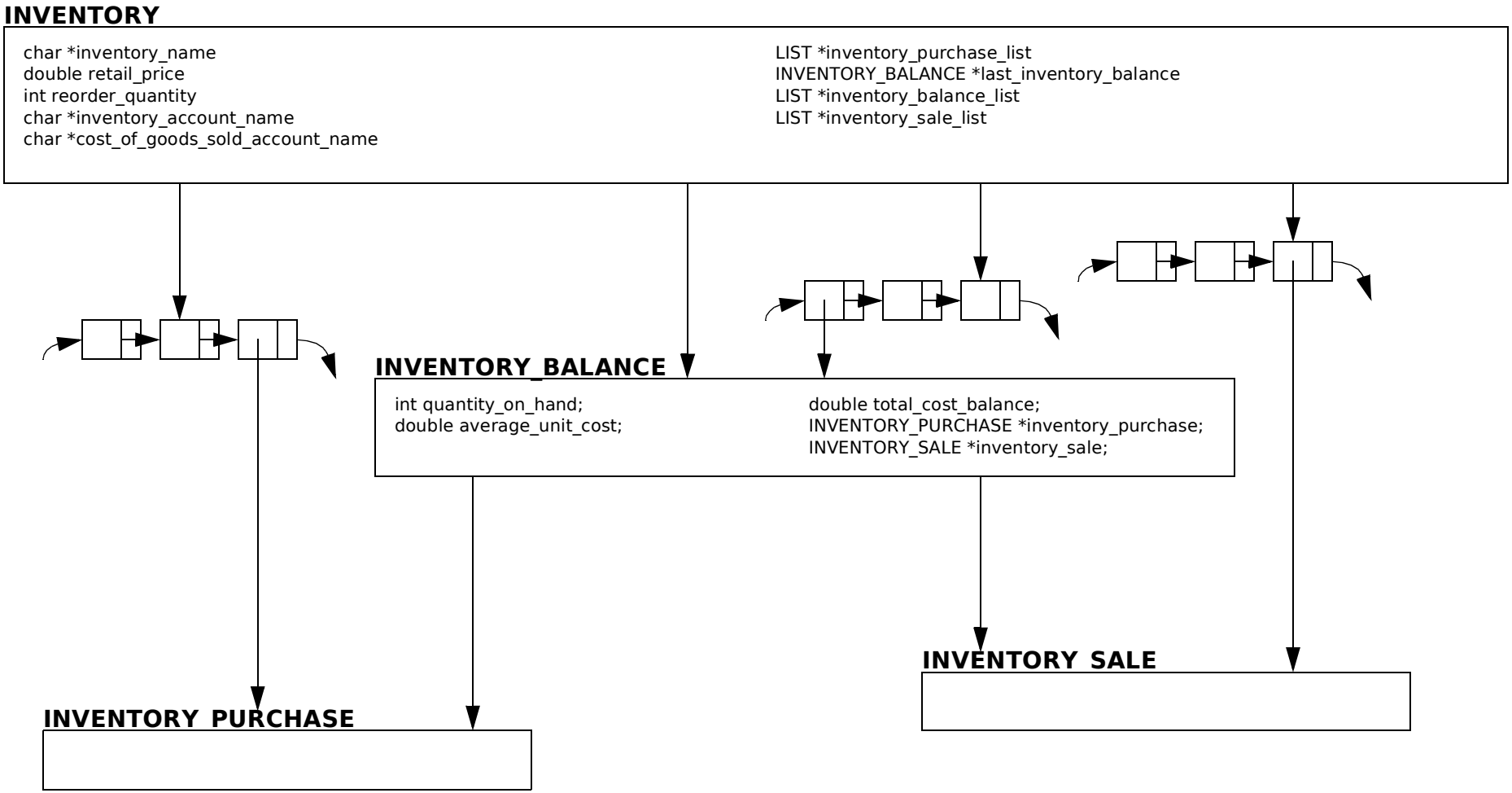
```
/* Usage */
double depreciation_amount(
    enum depreciation_method depreciation_method,
    char *service_placement_date,
    depreciation_prior_depreciation_date(),
    fixed_asset_purchase_depreciation_date(),
    double cost_basis,
    int depreciation_units_produced(),
    int estimated_residual_value,
    int estimated_useful_life_years,
    int estimated_useful_life_units,
    int declining_balance_n,
    double prior_accumulated_depreciation );

/* Process */
if ( depreciation_method == not_depreciated )
{
    double amount = 0.0;
}
else
if ( depreciation_method == straight_line )
{
    double amount =
      double depreciation_straight_line(
        char *service_placement_date,
        char *prior_depreciation_date,
        char *depreciation_date,
        double cost_basis,
        double estimated_residual_value,
        int estimated_useful_life_years,
        double prior_accumulated_depreciation );
}
else
if ( depreciation_method == sum_of_years_digits )
{
    double amount =
      double depreciation_sum_of_years_digits(
        char *service_placement_date,
        char *prior_depreciation_date,
        char *depreciation_date,
        double cost_basis,
        double estimated_residual_value,
        int estimated_useful_life_years,
        double prior_accumulated_depreciation );
}
else
if ( depreciation_method == double_declining_balance )
{
    double amount =
      depreciation_double_declining_balance(
        char *service_placement_date,
        char *prior_depreciation_date,
        char *depreciation_date,
        double cost_basis,
        double estimated_residual_value,
        int estimated_useful_life_years,
        double prior_accumulated_depreciation );
}
else
if ( depreciation_method == n_declining_balance )
{
    double amount =
      double depreciation_n_declining_balance(
        char *service_placement_date,
        char *prior_depreciation_date,
        char *depreciation_date,
        double cost_basis,
        double estimated_residual_value,
        int estimated_useful_life_years,
        int declining_balance_n,
        double prior_accumulated_depreciation );
}
else
if ( depreciation_method == units_of_production )
{
    double amount =
      double depreciation_units_of_production(
        double cost_basis,
        double estimated_residual_value,
        int estimated_useful_life_units,
        int units_produced,
        double prior_accumulated_depreciation );
}

if ( amount < 0.0 ) amount = 0.0;
```

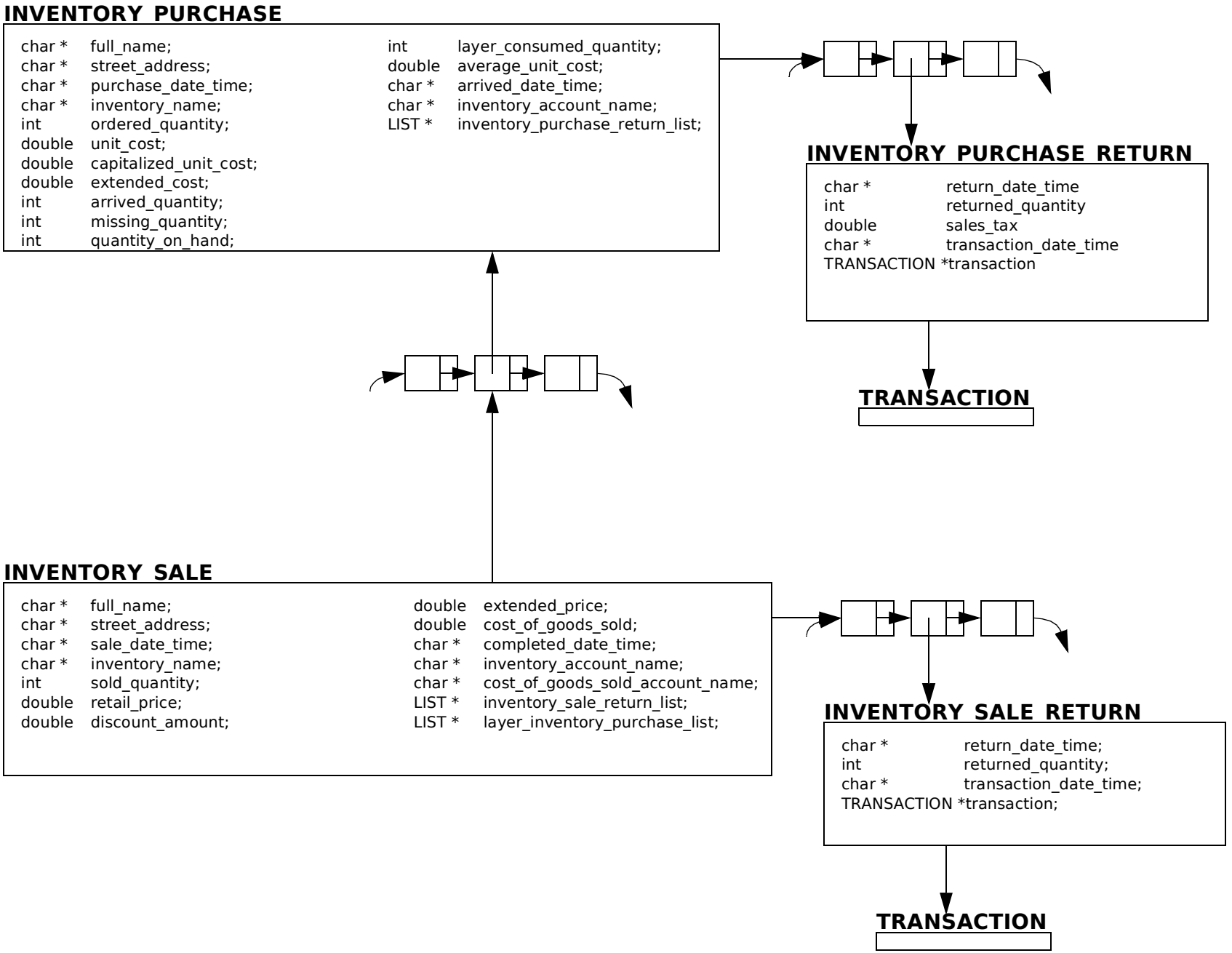
DEPRECIATION (3)

```
/* Usage */
int depreciation_units_produced(
    char *asset_name,
    char *serial_label,
    enum depreciation_method depreciation_method,
    int units_produced_so_far );

/* Process */
if ( depreciation_method != units_of_production )
{
    return 0;
}

int depreciation_units_produced_total(
    char *DEPRECIATION_TABLE,
    char *asset_name,
    char *serial_label );

int depreciation_units_produced_current(
    int units_produced_so_far,
    int depreciation_units_produced_total() );

return depreciation_units_produced_current();

/* Usage */
DEPRECIATION *depreciation_fetch(
    char *asset_name,
    char *serial_label,
    char *depreciation_date );

/* Process */
char *depreciation_primary_where(
    char *asset_name,
    char *serial_label,
    char *depreciation_date );

char *depreciation_system_string(
    char *DEPRECIATION_TABLE,
    char *where );

char *string_pipe_fetch(
    depreciation_system_string() );

DEPRECIATION *depreciation_parse(
    string_pipe_fetch()
      /* input */ );

/* Usage */
void depreciation_list_insert(
    LIST *fixed_asset_purchase_depreciation_list()
      /* depreciation_list */ );

/* Process */
FILE *depreciation_insert_pipe_open(
    char *DEPRECIATION_TABLE,
    char *DEPRECIATION_INSERT_COLUMNS );

for DEPRECIATION *depreciation in depreciation_list
{
    void depreciation_insert_pipe(
      char *depreciation->asset_name,
      char *depreciation->serial_label,
      char *depreciation->depreciation_date,
      int depreciation->units_produced,
      double depreciation->amount,
      char *depreciation->
        entity_self->
        entity->
        full_name,
      char *depreciation->
        entity_self->
        entity->
        street_address,
      char *transaction_date_time,
      FILE *depreciation_insert_pipe_open() );
}

void pclose( depreciation_insert_pipe_open() );

/* Usage */
void depreciation_update(
    int units_produced,
    double depreciation_amount,
    char *full_name,
    char *street_address,
    char *transaction_date_time,
    char *asset_name,
    char *serial_label,
    char *depreciation_date );

/* Process */
FILE *depreciation_update_pipe_open(
    char *DEPRECIATION_TABLE,
    char *DEPRECIATION_PRIMARY_KEY );

void depreciation_update_pipe(
    int units_produced,
    double depreciation_amount,
    char *full_name,
    char *street_address,
    char *transaction_date_time,
    char *fixed_asset_name_escape(
      asset_name ),
    char *serial_label,
    char *depreciation_date,
    FILE *depreciation_update_pipe_open() );

void pclose( depreciation_update_pipe_open() );
```

DEPRECIATION (4)

```
/* Usage */
LIST *depreciation_list_fetch(
    char *DEPRECIATION_TABLE,
    char *asset_name,
    char *serial_label );

/* Process */
char *fixed_asset_purchase_primary_where(
    asset_name,
    serial_label );

char *depreciation_system_string(
    DEPRECIATION_TABLE,
    fixed_asset_purchase_primary_where() );

LIST *depreciation_system_list(
    depreciation_system_string() );

/* Usage */
LIST *depreciation_system_list(
    char *depreciation_system_string() )

/* Process */
FILE *depreciation_input_pipe_open(
    char *depreciation_system_string );

for input[] in
    string_input(
      depreciation_input_pipe_open() )
{
    list_set(
      list,
      depreciation_parse( input ) );
}

void pclose( depreciation_input_pipe_open() );

/* Usage */
TRANSACTION *depreciation_transaction(
    char *entity_self->entity->full_name,
    char *entity_self->entity->street_address,
    char *fixed_asset_purchase_depreciation_date(),
    double depreciation_amount(),
    char *account_depreciation_string(),
    char *account_accumulated_depreciation_string() );

/* Process */
/* Increments second each invocation. */
char *transaction_increment_date_time(
    fixed_asset_purchase_depreciation_date );

/* Public */
enum depreciation_method
    depreciation_method_evaluate(
      depreciation_method_string );

char *depreciation_method_string(
    enum depreciation_method
      depreciation_method );

double depreciation_fraction_of_year(
    char *service_placement_date,
    char *prior_depreciation_date,
    char *depreciation_date_string );

char *depreciation_prior_depreciation_date(
    char *DEPRECIATION_TABLE,
    char *asset_name,
    char *serial_label,
    char *depreciation_date );

char *depreciation_subquery_exists_where(
    char *DEPRECIATION_TABLE,
    char *FIXED_ASSET_PURCHASE_TABLE,
    char *depreciation_date );

void depreciation_delete_open(
    char *DEPRECIATION_PRIMARY_KEY,
    char *DEPRECIATION_TABLE );
```


INVENTORY
char *inventory_name
double retail_price
int reorder_quantity
char *inventory_account_name
char *cost_of_goods_sold_account_name
LIST *inventory_purchase_list
INVENTORY_BALANCE *last_inventory_balance
LIST *inventory_balance_list
LIST *inventory_sale_list
INVENTORY_BALANCE
int quantity_on_hand;
double average_unit_cost;
double total_cost_balance;
INVENTORY_PURCHASE *inventory_purchase;
INVENTORY_SALE *inventory_sale;
INVENTORY_SALE
INVENTORY_PURCHASE


INVENTORY_PURCHASE
char * full_name;
char * street_address;
char * purchase_date_time;
char * inventory_name;
int ordered_quantity;
double unit_cost;
double capitalized_unit_cost;
double extended_cost;
int arrived_quantity;
int missing_quantity;
int quantity_on_hand;
int layer_consumed_quantity;
double average_unit_cost;
char * arrived_date_time;
char * inventory_account_name;
LIST * inventory_purchase_return_list;
INVENTORY_PURCHASE_RETURN
char * return_date_time
int returned_quantity
double sales_tax
char * transaction_date_time
TRANSACTION *transaction
TRANSACTION
INVENTORY_SALE
char * full_name;
char * street_address;
char * sale_date_time;
char * inventory_name;
int sold_quantity;
double retail_price;
double discount_amount;
double extended_price;
double cost_of_goods_sold;
char * completed_date_time;
char * inventory_account_name;
char * cost_of_goods_sold_account_name;
LIST * inventory_sale_return_list;
LIST * layer_inventory_purchase_list;
INVENTORY_SALE_RETURN
char * return_date_time;
int returned_quantity;
char * transaction_date_time;
TRANSACTION *transaction;
TRANSACTION

INVENTORY_SALE (1)

```
/* Usage */
LIST *inventory_sale_list(
    const char *INVENTORY_SALE_TABLE,
    char *full_name,
    char *street_address,
    char *sale_date_time );

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

char *sale_primary_where(
    full_name,
    street_address,
    sale_date_time );

char *appaserver_system_string(
    INVENTORY_SELECT,
    INVENTORY_SALE_TABLE,
    sale_primary_where() );

FILE *appaserver_input_pipe(
    appaserver_system_string() );

free( appaserver_system_string() );

while ( string_input( input[], appaserver_input_pipe() )
{
    INVENTORY_SALE *inventory_sale_parse(
        full_name,
        street_address,
        sale_date_time,
        input );

    list_set(
        list,
        inventory_sale_parse() );
}

/* Usage */
INVENTORY_SALE *inventory_sale_parse(
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *input );

double SALE_EXTENDED_PRICE(
    this->retail_price,
    this->quantity,
    this->discount_amount );

/* Process */
INVENTORY_SALE *inventory_sale_new(
    full_name,
    street_address,
    sale_date_time,
    strdup( inventory_name ) );

/* Usage */
INVENTORY_SALE *inventory_sale_new(
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *inventory_name );

/* Process */
INVENTORY_SALE *inventory_sale_calloc(
    void );

/* Attributes */
char *full_name;
char *street_address;
char *sale_date_time;
char *inventory_name;
int quantity;
double retail_price;
double discount_amount;
double extended_price;
double cost_of_goods_sold;
double sale_extended_price;
```

INVENTORY_SALE (2)

```
/* Usage */
double inventory_sale_total(
    LIST *inventory_sale_list );

/* Process */
for INVENTORY_SALE *inventory_sale in
    inventory_sale_list
{
    double total += inventory_sale->extended_price;
}

/* Usage */
double inventory_sale_CGS_total(
    LIST *inventory_sale_list );

/* Process */
for INVENTORY_SALE *inventory_sale in
    inventory_sale_list
{
    double total += inventory_sale->cost_of_goods_sold;
}

/* Usage */
INVENTORY_SALE *inventory_sale_seek(
    LIST *inventory_sale_list,
    char *inventory_name );

/* Process */

/* Usage */
void inventory_sale_update(
    const char *INVENTORY_SALE_TABLE,
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *inventory_name,
    double SALE_EXTENDED_PRICE );

/* Process */
char *inventory_sale_update_system_string(
    const char *INVENTORY_SALE_TABLE );

FILE *appaserver_output_pipe(
    inventory_sale_update_system_string() );

free( inventory_sale_update_system_string() );

fprintf(
    appaserver_output_pipe(),
    "%s^%s^%s^%s^extended_price^%.2lf\n",
    full_name,
    street_address,
    sale_date_time,
    inventory_name,
    SALE_EXTENDED_PRICE );

pclose( appaserver_output_pipe() );
```

SPECIFIC_INVENTORY_SALE (1)

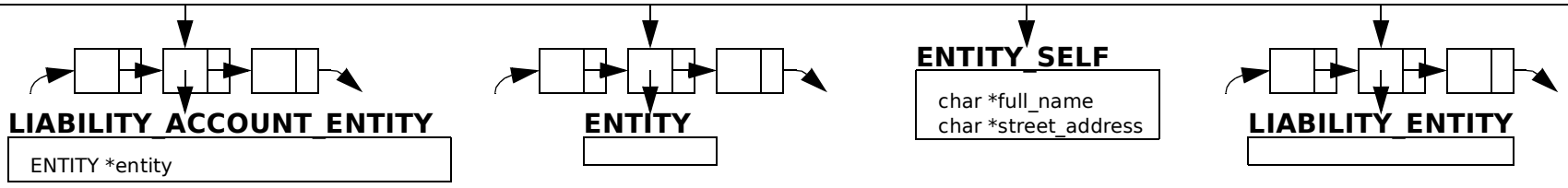
```
/* Usage */
LIST *specific_inventory_sale_list(
    const char *SPECIFIC_INVENTORY_SALE_TABLE,
    char *full_name,
    char *street_address,
    char *sale_date_time );

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

char *sale_primary_where(
    full_name,
    street_address,
    sale_date_time );

char *appaserver_system_string(
    SPECIFIC_INVENTORY_SELECT,
    SPECIFIC_INVENTORY_SALE_TABLE,
    sale_primary_where() );

FILE *appaserver_input_pipe(
    appaserver_system_string() );

free( appaserver_system_string() );

while ( string_input( input[], appaserver_input_pipe() ) )
{
    SPECIFIC_INVENTORY_SALE *
       specific_inventory_sale_parse(
          full_name,
          street_address,
          sale_date_time,
          input );

    list_set(
       list,
       specific_inventory_sale_parse() );
}

/* Usage */
SPECIFIC_INVENTORY_SALE *
    specific_inventory_sale_parse(
       char *full_name,
       char *street_address,
       char *sale_date_time,
       char *input );

double SALE_EXTENDED_PRICE(
    this->retail_price,
    1 /* quantity */,
    this->discount_amount );

/* Process */
SPECIFIC_INVENTORY_SALE *
    specific_inventory_sale_new(
       full_name,
       street_address,
       sale_date_time,
       strdup( inventory_name ),
       strdup( serial_label ) );

/* Usage */
SPECIFIC_INVENTORY_SALE *
    specific_inventory_sale_new(
       char *full_name,
       char *street_address,
       char *sale_date_time,
       char *inventory_name,
       char *serial_label );

/* Process */
SPECIFIC_INVENTORY_SALE *
    specific_inventory_sale_calloc(
       void );

/* Attributes */
char *full_name;
char *street_address;
char *sale_date_time;
char *inventory_name;
char *serial_label;
double retail_price;
double discount_amount;
double extended_price;
double cost_of_goods_sold;
double sale_extended_price;
```

SPECIFIC_INVENTORY_SALE (2)

```
/* Usage */
double specific_inventory_sale_total(
    LIST *specific_inventory_sale_list );

/* Process */
for SPECIFIC_INVENTORY_SALE *specific_inventory_sale
    in specific_inventory_sale_list
{
    double total +=
        specific_inventory_sale->
            extended_price;
}

/* Usage */
double specific_inventory_sale_CGS_total(
    LIST *specific_inventory_sale_list );

/* Process */
for SPECIFIC_INVENTORY_SALE *specific_inventory_sale
    in specific_inventory_sale_list
{
    double total +=
        specific_inventory_sale->
            cost_of_goods_sold;
}

/* Usage */
SPECIFIC_INVENTORY_SALE *
    specific_inventory_sale_seek(
        LIST *specific_inventory_sale_list,
        char *inventory_name,
        char *serial_label );

/* Process */

/* Usage */
void specific_inventory_sale_update(
    const char *SPECIFIC_INVENTORY_SALE_TABLE,
    char *full_name,
    char *street_address,
    char *sale_date_time,
    char *inventory_name,
    char *serial_label,
    double sale_extended_price );

/* Process */
char *specific_inventory_sale_update_system_string(
    const char *SPECIFIC_INVENTORY_SALE_TABLE );

FILE *appaserver_output_pipe(
    specific_inventory_sale_update_system_string() );

free( specific_inventory_sale_update_system_string() );

fprintf(
    appaserver_output_pipe(),
    "%s^%s^%s^%s^%s^extended_price^%.2lf\n",
    full_name,
    street_address,
    sale_date_time,
    inventory_name,
    serial_label,
    sale_extended_price );

pclose( appaserver_output_pipe() );
```

LIABILITY_CALCULATE

```
/* Usage */
LIABILITY_CALCULATE *liability_calculate_new(
    char *application_name );

/* Process */
LIABILITY_CALCULATE *liability_calculate_calloc( void );

LIST *liability_account_entity_list();

LIST *exclude_account_name_list =
     LIST *liability_account_entity_account_name_list(
        liability_account_entity_list() );

list_set(
     exclude_account_name_list,
     ACCOUNT_UNCLEARED_CHECKS );

LIST *account_current_liability_name_list(
    ACCOUNT_TABLE,
    SUBCLASSIFICATION_CURRENT_LIABILITY,
    ACCOUNT_CREDIT_CARD_KEY,
    exclude_account_name_list );

LIST *journal_account_distinct_entity_list(
    JOURNAL_TABLE,
    account_current_liability_name_list()
       /* account_name_list */ );

LIST *account_receivable_name_list(
    ACCOUNT_TABLE,
    SUBCLASSIFICATION_RECEIVABLE );

ENTITY_SELF *entity_self =
    ENTITY_SELF *entity_self_fetch(
       0 /* not fetch_entity_boolean */ );

LIST *liability_entity_list = list_new();

if ( list_length( liability_account_entity_list() )
{
    list_set_list(
       liability_entity_list,
       liability_entity_list_account(
          liability_account_entity_list() );
}

if ( list_length( journal_account_distinct_entity_list() )
{
    list_set_list(
       liability_entity_list,
       liability_entity_list_entity(
          account_current_liability_name_list(),
          journal_account_distinct_entity_list(),
          account_receivable_name_list(),
          entity_self );
}

if ( !list_length( liability_entity_list ) )
{
    return (LIABILITY_CALCULATE *)0;
}

/* Driver */
void liability_calculate_stdout(
    LIST *liability_entity_list );

/* Process */
for LIABILITY_ENTITY *liability_entity in liability_entity_list
{
    if ( liability_entity->amount_due > 0.0 )
    {
       printf("%s\n",
             liability_entity_display(
                liability_entity );
    }
}

/* Attributes */
LIST *liability_account_entity_list;
LIST *exclude_account_name_list;
LIST *account_current_liability_name_list;
LIST *journal_account_distinct_entity_list;
LIST *account_receivable_name_list;
ENTITY_SELF *entity_self;
LIST *liability_entity_list;
```

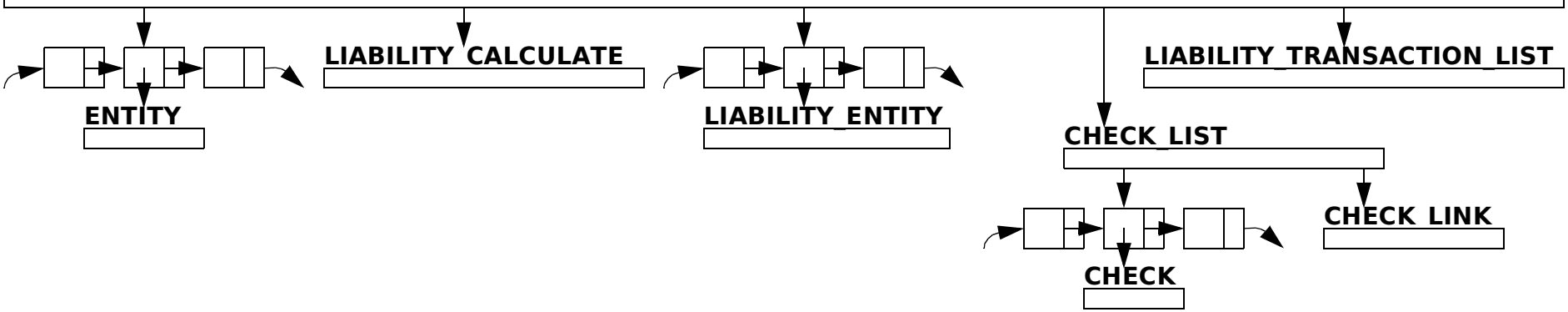


LIABILITY_PAYMENT (1)

```
/* Usage */
LIABILITY_PAYMENT *liability_payment_new(
    char *application_name,
    char *cash_account_name,
    double dialog_box_payment_amount,
    int starting_check_number,
    char *dialog_box_memo,
    char *appaserver_parameter_data_directory,
    char *process_name,
    char *session_key,
    LIST *entity_full_street_list(
       full_name_list,
       street_address_list ) );

/* Process */
LIABILITY_PAYMENT *liability_payment_calloc( void );

if ( !list_length( entity_full_street_list() )
{
   char *liability_payment_error_message();
   return this;
}

char *transaction_memo( dialog_box_memo );

if (   list_length( entity_full_street_list() > 1
&&   (  dialogy_box_payment_amount
||        *transaction_memo() ) )
{
   char *liability_payment_error_message();
   return this;
}

LIABILITY_CALCULATE *liability_calculate =
   liability_calculate_new(
      application_name );

if ( !liability_calculate )
{
   char *liability_payment_error_message();
   return this;
}

LIST *liability_payment_entity_list(
   entity_full_street_list(),
   liability_calculate->liability_entity_list );

if ( !list_length( liability_payment_entity_list() ) )
{
   char *liability_payment_error_message();
   return this;
}
```

LIABILITY_PAYMENT (2)

```
/* Process (continued) */
if ( starting_check_number )
{
   CHECK_LIST *check_list =
      check_list_new(
         application_name,
         dialog_box_payment_amount,
         starting_check_number,
         transaction_memo(),
         appaserver_parameter_data_directory,
         process_name,
         session_key,
         liability_payment_entity_list() );
}

char *liability_payment_credit_account_name(
   int starting_check_number,
   char *cash_account_name,
   char *account_uncleared_checks_string() );

LIABILITY_TRANSACTION_LIST *liability_transaction_list =
   liability_transaction_list_new(
      dialog_box_payment_amount,
      starting_check_number,
      transaction_memo(),
      liability_payment_entity_list(),
      liability_payment_credit_account_name() );

if ( !liability_transaction_list )
{
   char *liability_payment_error_message();
   return this;
}

/* Usage */
LIST *liability_payment_entity_list(
   LIST *entity_full_street_list(),
   LIST *liability_calculate->liability_entity_list );

/* Process */
for ENTITY *entity in entity_full_street_list()
{
   LIABILITY_ENTITY *liability_entity =
      liability_entity_seek(
         entity->full_name,
         entity->street_address,
         liability_entity_list );

   list_set(
      entity_list,
      liability_entity );
}

/* Public */
char *liability_payment_error_message(
   char *message );

/* Driver */
if ( liability_payment->liability_check_list )
{
   printf(
      "%s\n",
      liability_payment->
         liability_check_list->
         liability_check_link->
         pdf_anchor_html );

   printf( "<p>\n" );
}

if ( execute )
{
    /* May reset transaction_date_time */
    void liability_transaction_list_insert(
       liability_payment->liability_transaction_list );
}

void liability_transaction_list_html_display(
    liability_payment->liability_transaction_list );

/* Attributes */
LIST *entity_list;
LIABILITY_CALCULATE *liability_calculate;
LIST *liability_payment_entity_list;
char *transaction_memo;
CHECK_LIST *liability_check_list;
char *credit_account_name;
LIABILITY_TRANSACTION_LIST *liability_transaction_list;
char *error_message;
```

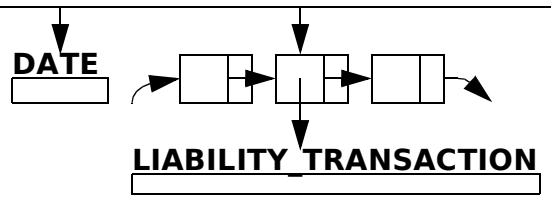


LIABILITY_ACCOUNT_ENTITY

```
/* Usage */
LIST *liability_account_entity_list( void );

/* Process */
char *liability_account_entity_system_string(
    char *LIABILITY_ACCOUNT_ENTITY_SELECT,
    char *LIABILITY_ACCOUNT_ENTITY_TABLE );

LIST *liability_account_entity_system_list(
    liability_account_entity_system_string() );

/* Usage */
LIST *liability_account_entity_system_list(
    char *liability_account_entity_system_string() );

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

FILE *appaserver_input_pipe(
    liability_account_entity_system_string );

while ( string_input( appaserver_input_pipe() ) )
{
    LIABILITY_ACCOUNT_ENTITY *
        liability_account_entity_parse(
            string_input() );

    list_set(
        list,
        liability_account_entity_parse() );
}
pclose( appaserver_input_pipe() );

/* Usage */
LIABILITY_ACCOUNT_ENTITY *
    liability_account_entity_parse(
        char *string_input );

/* Process */
LIABILITY_ACCOUNT_ENTITY *
    liability_account_entity_new(
        strdup( account_name );

/* Usage */
LIABILITY_ACCOUNT_ENTITY *
    liability_account_entity_new(
        char *account_name );

/* Process */
LIABILITY_ACCOUNT_ENTITY *
    liability_account_entity_calloc(
        void );

ENTITY *entity =
    entity_new(
        strdup( full_name ),
        strdup( street_address ) );

/* Public */
LIABILITY_ACCOUNT_ENTITY *
    liability_account_entity_seek(
        char *account_name,
        LIST *liability_account_entity_list() );

LIST *liability_account_entity_account_name_list(
    LIST *liability_account_entity_list() );

/* Attributes */
char *account_name;
ENTITY *entity;
```

LIABILITY_TRANSACTION_LIST

```
/* Usage */
LIABILITY_TRANSACTION_LIST *
   liability_transaction_list_new(
      double dialog_box_payment_amount,
      int starting_check_number,
      char *transaction_memo(),
      LIST *liability_payment_entity_list,
      char *liability_payment_credit_account_name );

/* Process */
if ( dialog_box_payment_amount
&& list_length( liability_payment_entity_list ) > 1 )
{
   return NULL;
}

LIABILITY_TRANSACTION_LIST *
   liability_transaction_list_calloc(
      void );

LIST *list = list_new();

DATE *transaction_date_time =
   date_now_new(
      date_utc_offset() );

for LIABILITY_ENTITY *liability_entity in
   liability_payment_entity_list
{
   list_set(
      list,
      liability_transaction_new(
         dialog_box_payment_amount,
         starting_check_number
            /* check_number */,
         transaction_memo,
         liability_payment_credit_account_name,
         liability_entity,
         transaction_date_time ) );

   if ( starting_check_number )
   {
      starting_check_number++;
   }

   date_increment_seconds( transaction_date_time, 1 );
}

/* Usage */
/* May reset transaction_date_time */
void liability_transaction_list_insert(
   LIABILITY_TRANSACTION_LIST *
      liability_transaction_list,
   char *appaserver_error_filename );

/* Process */
LIST *liability_transaction_list_extract(
   liability_transaction_list->list );

char *transaction_list_insert(
   liability_transaction_list_extract(),
   1 /* insert_journal_list_boolean */,
   0 /* not transaction_lock_boolean */ );

/* Usage */
void liability_transaction_list_html_display(
   LIABILITY_TRANSACTION_LIST *
      liability_transaction_list );

/* Process */
LIST *liability_transaction_list_extract(
   liability_transaction_list->list );

void transaction_list_html_display(
   liability_transaction_list_extract() );

/* Public */
LIST *liability_transaction_list_extract(
   LIST *liability_transaction_list->list );

/* Attributes */
DATE *transaction_date_time;
LIST *list;
```



LIABILITY_TRANSACTION

```
/* Usage */
LIABILITY_TRANSACTION *liability_transaction_new(
    double dialog_box_payment_amount,
    int check_number,
    char *transaction_memo(),
    char *liability_payment_credit_account_name,
    LIABILITY_ENTITY *liability_entity,
    DATE *transaction_date_time );

/* Process */
LIABILITY_TRANSACTION *liability_transaction_calloc(
    void );
```

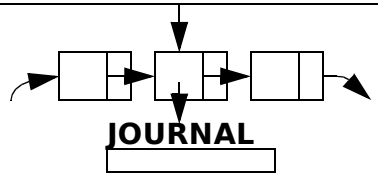
```
LIABILITY_JOURNAL_LIST *liability_journal_list =
    liability_journal_list_new(
        dialog_box_payment_amount,
        liability_payment_credit_account_name,
        liability_entity );

if ( !liability_journal_list ) return NULL;

TRANSACTION *transaction =
    transaction_entity_new(
        liability_entity->entity,
        date_display_19( transaction_date_time ),
        liability_journal_list->transaction_amount,
        check_number,
        (*transaction_memo)
            ? transaction_memo
            : LIABILITY_TRANSACTION_MEMO,
        liability_journal_list->journal_list );
```

```
/* Attributes */
LIABILITY_JOURNAL_LIST *liability_journal_list;
TRANSACTION *transaction;

/* Set externally */
double journal_list_transaction_amount;
```

LIABILITY_JOURNAL_LIST **TRANSACTION**

LIABILITY_JOURNAL_LIST

```
/* Usage */
LIABILITY_JOURNAL_LIST *liability_journal_list_new(
    double dialog_box_payment_amount,
    char *liability_payment_credit_account_name,
    LIABILITY_ENTITY *liability_entity );

/* Process */
LIABILITY_JOURNAL_LIST *liability_journal_list_calloc(
    void );

LIST *journal_list = list_new();

double liability_journal_list_transaction_amount(
    double dialog_box_payment_amount,
    double liability_entity->amount_due );

list_set(
    journal_list,
        journal_credit_new(
            liability_payment_credit_account_name,
            liability_journal_list_transaction_amount() ) );

for LIABILITY_ACCOUNT *liability_account in
    liability_entity->liability->liability_account_list->list
{
    list_set(
        journal_list,
        journal_debit_new(
            liability_account->account_name,
            (dialog_box_payment_amount)
                ? dialog_box_payment_amount
                : liability_account->credit_amount ) );
}

if ( liability_entity->receivable )
{
    for RECEIVABLE_ACCOUNT *receivable_account in
        liability_entity->
            receivable->
            receivable_account_list->
            list
    {
        list_set(
            journal_list,
            journal_credit_new(
                receivable_account->account_name,
                receivable_account->debit_amount ) );
    }
}

/* Attributes */
LIST *journal_list;
double transaction_amount;
```



LIABILITY_ENTITY (1)

```
/* Usage */
LIST *liability_entity_list_account(
    LIST *liability_account_entity_list() );

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

for LIABILITY_ACCOUNT_ENTITY *liability_account_entity
    in liability_account_entity_list
{
    LIABILITY_ENTITY *liability_entity =
        liability_entity_account_name_new(
            liability_account_entity->account_name,
            liability_account_entity->entity );

    if ( liability_entity )
    {
        liability_entity->entity =
            liability_account_entity->entity;

        list_set(
            liability_entity_list,
            liability_entity );
    }
}

/* Usage */
LIST *liability_entity_list_entity(
    LIST *account_current_liability_name_list(),
    LIST *journal_account_distinct_entity_list(),
    LIST *account_receivable_name_list(),
    ENTITY_SELF *entity_self );

/* Process */
for ENTITY *entity in journal_account_distinct_entity_list    }
{
if (   strcmp(
           entity->full_name,
           entity_self->full_name ) == 0
&&  strcmp(
           entity->street_address,
           entity_self->street_address ) == 0 )
{
   continue;
}

list_set(
   liability_entity_list,
   liability_entity_account_list_new(
      account_current_liability_name_list,
      account_receivable_name_list,
      entity ) );
```

LIABILITY_ENTITY (2)

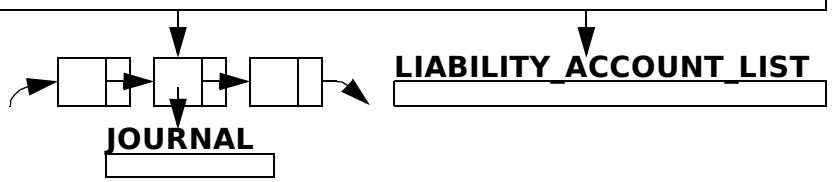
```
/* Usage */
LIABILITY_ENTITY *
    liability_entity_account_name_new(
        char *account_name,
        ENTITY *entity );

/* Process */
LIABILITY_ENTITY *liability_entity_calloc(
    void );

LIABILITY *liability =
    liability_account_fetch(
        account_name );

if ( !liability ) return NULL;

double liability_entity_amount_due(
    liability,
    (RECEIVABLE *receivable );

/* Usage */
LIABILITY_ENTITY *liability_entity_account_list_new(
    LIST *account_current_liability_name_list(),
    LIST *account_receivable_name_list(),
    ENTITY *entity );

/* Process */
```

```
LIABILITY_ENTITY *liability_entity_calloc();

LIABILITY *liability =
    liability_entity_fetch(
        entity->full_name,
        entity->street_address,
        account_current_liability_name_list );

if ( !liability ) return NULL;

RECEIVABLE *receivable =
    receivable_fetch(
        entity->full_name,
        entity->street_address,
        account_receivable_name_list );

double liability_entity_amount_due(
    liability,
    receivable );

/* Attributes */
ENTITY *entity;
LIABILITY *liability;
RECEIVABLE *receivable;
double amount_due;

/* Usage */
```

```
double liability_entity_amount_due(
    LIABILITY *liability,
    RECEIVABLE *receivable );

/* Process */

/* Usage */
LIABILITY_ENTITY *liability_entity_seek(
    char *full_name,
    char *street_address,
    LIST *liability_calculate->liability_entity_list );

/* Process */

/* Usage */
char *liability_entity_display(
    LIABILITY_ENTITY *liability_entity );

/* Process */

/* Attributes */
ENTITY *entity;
LIABILITY *liability;
RECEIVABLE *receivable;
double amount_due;
```

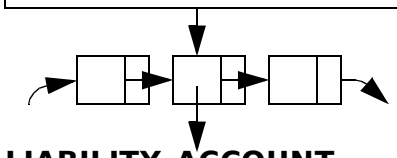
LIABILITY

```
/* Usage */
LIABILITY *liability_account_fetch(
    char *liability_account_name );

/* Process */
LIABILITY *liability_calloc(
    void );

char *liability_account_where(
    char *liability_account_name );

LIST *journal_system_list(
    journal_system_string(
        JOURNAL_SELECT,
        JOURNAL_TABLE,
        liability_account_where() ),
    0 /* not fetch_account */,
    0 /* not fetch_subclassification */,
    0 /* not fetch_element */,
    0 /* not fetch_transaction */ );

if ( !list_length( journal_system_list() ) return NULL;

double journal_credit_debit_difference_sum(
    journal_system_list() );

if ( !journal_credit_debit_difference_sum() ) return NULL;

LIABILITY_ACCOUNT_LIST *
    liability_account_list =
        liability_account_list_new(
            journal_system_list() );

/* Usage */
LIABILITY *liability_entity_fetch(
    char *full_name,
    char *street_address,
    LIST *account_current_liability_name_list() );

/* Process */
LIABILITY *liability_calloc();

char *string_in_clause(
    account_current_liability_name_list
        /* data_list */ );

char *liability_entity_where(
    char *full_name,
    char *street_address,
    char *string_in_clause() );

LIST *journal_system_list(
    journal_system_string(
        JOURNAL_SELECT,
        JOURNAL_TABLE,
        liability_entity_where() ),
    0 /* not fetch_account */,
    0 /* not fetch_subclassification */,
    0 /* not fetch_element */,
    0 /* not fetch_transaction */ );

if ( !list_length( journal_system_list() ) return NULL;

double journal_credit_debit_difference_sum(
    journal_system_list() );

if (!journal_credit_debit_difference_sum()
||  journal_credit_debit_difference_sum() < 0.0 )
{
    return NULL;
}

LIABILITY_ACCOUNT_LIST *
    liability_account_list =
        liability_account_list_new(
            journal_system_list() );

/* Attributes */
char *account_where;
LIST *journal_system_list;
double journal_credit_debit_difference_sum;
char *string_in_clause;
char *entity_where;
LIABILITY_ACCOUNT_LIST *liability_account_list;
```



LIABILITY_ACCOUNT_LIST

```
/* Usage */
LIABILITY_ACCOUNT_LIST *
    liability_account_list_new(
       LIST *journal_system_list() );

/* Process */
LIABILITY_ACCOUNT_LIST *liability_account_list_calloc(
    void );

LIST *list = list_new();

for JOURNAL *journal in journal_system_list
{
    LIABILITY_ACCOUNT *
       liability_account =
          liability_account_getset(
             list,
             journal->account_name );

    if ( journal->credit_amount )
    {
       liability_account->credit_amount +=
          journal->credit_amount;
    }
    else
    {
       liability_account->credit_amount -=
          journal->debit_amount;
    }
}

/* Attribute */
LIST *list;
```

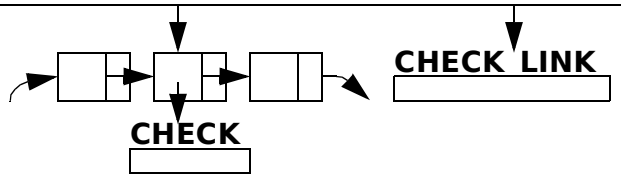


LIABILITY_ACCOUNT

```
/* Usage */
LIABILITY_ACCOUNT *liability_account_getset(
    LIST *list,
    char *account_name );

/* Process */
for LIABILITY_ACCOUNT *liability_account in list
{
    if ( strcmp(
       account_name,
       liability_account->account_name ) == 0 )
    {
       return liability_account;
    }
}

LIABILITY_ACCOUNT *liability_account =
    liability_account_new(
       account_name );

list_set( list, liability_account );

/* Usage */
LIABILITY_ACCOUNT *liability_account_new(
    char *account_name );

/* Process */
LIABILITY_ACCOUNT *liability_account_calloc(
    void );

/* Attributes */
char *account_name;
double credit_amount;
```

CHECK_LIST

```
/* Usage */
CHECK_LIST *check_list_new(
    char *application_name,
    double dialog_box_payment_amount,
    int starting_check_number,
    char *transaction_memo(),
    char *appaserver_parameter_data_directory,
    char *process_name,
    char *session_key,
    LIST *liability_payment_entity_list() );

/* Process */
CHECK_LIST *check_list_calloc(
    void );

char *check_list_documentclass(
    void );

char *check_list_usepackage(
    void );

char *check_list_pagenumbering_gobble(
    void );

char *check_list_begin_document(
    void );

char *check_list_heading(
    char *check_list_documentclass(),
    char *check_list_usepackage(),
    char *check_list_pagenumbering_gobble(),
    char *check_list_begin_document() );

LIST *list = list_new();

for LIABILITY_ENTITY *liability_entity in
    liability_payment_enitity_list
{
    list_set(
        list,
        check_new(
            dialog_box_payment_amount,
            starting_check_number++
                /* check_number */,
            transaction_memo,
            liability_entity->amount_due,
            liabilility_entity->entity->full_name ) );
}

char *check_list_end_document(
    void );

CHECK_LINK *check_link =
    check_link_new(
        application_name,
        process_name,
        appaserver_parameter_data_directory,
        session_key );

char *latex_directory_filename(
    check_link->tex_filename,
    checklink_appaserver_link_working_directory );

FILE *appaserver_output_file(
    latex_directory_filename() );

fprintf(
    appaserver_output_file(),
    "%s\n",
    check_list_heading() );

for CHECK *check in list
{
    fprintf(
        appaserver_output_file(),
        "%s\n",
        liability_check->output_string );
}

fprintf(
    appaserver_output_file(),
    "%s\n",
    check_list_end_document() );

fclose( appaserver_output_file() );

void latex_tex2pdf(
    check_link->
        tex_filename,
    check_link->
        appaserver_link_working_directory );

/* Attributes */
char *document_class;
char *usepackage;
char *pagenumbering_gobble;
char *begin_document;
char *heading;
LIST *list;
char *end_document;
CHECK_LINK *check_link;
```

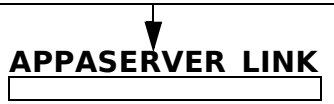


CHECK

```
/* Usage */
CHECK *check_new(
    double dialog_box_payment_amount,
    int check_number,
    char *transaction_memo(),
    double liability_entity_amount_due,
    char *entity_full_name );

/* Process */
CHECK *check_calloc(
    void );

double check_amount(
    double dialog_box_payment_amount,
    double liability_entity_amount_due );

char *check_dollar_text(
    double check_amount() );

char *check_escape_payable_to(
    char *entity_full_name );

char *check_move_down( void );

char *check_date_display(
    char *string_pipe_fetch(
        char *CHECK_DATE_COMMAND )
            /* check_date */ );

char *string_commas_money(
    double check_amount() );

char *check_vendor_name_amount_due_display(
    char *check_escape_payable_to(),
    char *string_commas_money() );

char *check_amount_due_stub_display(
    char *string_commas_money() );

char *check_dollar_text_display(
    char *check_dollar_text() );

char *check_memo_display(
    transaction_memo );

char *check_number_display(
    int check_number );

char *check_newpage(
    void );

char *check_output_string(
    char *check_move_down(),
    char *check_date_display(),
    char *check_vendor_name_amount_due_display(),
    char *check_amount_due_stub_display(),
    char *check_dollar_text_display(),
    char *check_memo_display(),
    char *check_number_display(),
    char *check_newpage() );

/* Usage */
char *check_memo_display(
    char *transaction_memo );

/* Process */
char *check_stub_memo(
    char *transaction_memo,
    30 /* int max_length */ );

/* Attributes */
double amount;
char *dollar_text;
char *escape_payable_to;
char *move_down;
char *date_display;
char *vendor_name_amount_due_display;
char *amount_due_stub_display;
char *dollar_text_display;
char *memo_display;
char *number_display;
char *newpage;
char *output_string;
```

CHECK_LINK

```
/* Usage */
CHECK_LINK *check_link_new(
    char *application_name,
    char *process_name,
    char *appaserver_parameter_data_directory,
    char *session_key );

/* Process */
CHECK_LINK *check_link_calloc(
    void );

APPASERVER_LINK *appaserver_link =
    APPASERVER_LINK *appaserver_link_new(
       application_http_prefix(
          application_ssl_support_boolean(
             application_name ) ),
       application_server_address(),
       appaserver_parameter_data_directory,
       process_name /* filename_stem */,
       application_name,
       (pid_t)0 /* process_id */,
       session_key,
       (char *)0 /* begin_date_string */,
       (char *)0 /* end_date_string */,
          "tex" /* extension */ );

char *appaserver_link->extension = "tex";

char *tex_filename =
    char *appaserver_link_filename(
       appaserver_link->filename_stem,
       appaserver_link->begin_date_string,
       appaserver_link->end_date_string,
       appaserver_link->process_id,
       appaserver_link->session_key,
       appaserver_link->extension );

char *appaserver_link->extension = "pdf";

APPASERVER_LINK_PROMPT *
    appaserver_link->appaserver_link_prompt =
       appaserver_link_prompt_new(
          APPASERVER_LINK_PROMPT_DIRECTORY
             /* probably appaserver_data */,
          appaserver_link->http_prefix,
          appaserver_link->server_address,
          application_name,
          appaserver_link->filename_stem,
          appaserver_link->begin_date_string,
          appaserver_link->end_date_string,
          appaserver_link->process_id,
          appaserver_link->session_key,
          appaserver_link->extension );

char *pdf_anchor_html =
    char *appaserver_link_anchor_html(
       appaserver_link->
          appaserver_link_prompt->
          filename /* prompt_filename */,
       process_name /* target_window */,
       CHECK_PROMPT );

char *appaserver_link_working_directory(
    appaserver_parameter_data_directory,
    application_name );

/* Attributes */
APPASERVER_LINK *appaserver_link;
char *tex_filename;
char *pdf_anchor_html;
char *appaserver_link_working_directory;
```



RECEIVABLE

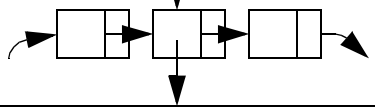
```
/* Usage */
RECEIVABLE *receivable_fetch(
    char *full_name,
    char *street_address,
    LIST *account_receivable_name_list() );

/* Process */
RECEIVABLE *receivable_calloc(
    void );

char *string_in_clause(
    account_receivable_name_list
       /* data_list */ );

char *receivable_where(
    char *full_name,
    char *street_address,
    char *string_in_clause() );

LIST *journal_system_list(
    journal_system_string(
       JOURNAL_SELECT,
       JOURNAL_TABLE,
       receivable_where() ),
    0 /* not fetch_account */,
    0 /* not fetch_subclassification */,
    0 /* not fetch_element */,
    0 /* not fetch_transaction */ );

if ( !list_length( journal_system_list() ) return NULL;

double journal_debit_credit_difference_sum(
    journal_system_list() );

if (   !journal_debit_credit_difference_sum()
||     journal_debit_credit_difference_sum() < 0.0 )
{
    return NULL;
}

RECEIVABLE_ACCOUNT_LIST *
    receivable_account_list =
       receivable_account_list_new(
          journal_system_list() );

/* Attributes */
LIST *journal_system_list;
double journal_debit_credit_difference_sum;
RECEIVABLE_ACCOUNT_LIST *receivable_account_list;
```

RECEIVABLE_ACCOUNT_LIST

```
/* Usage */
RECEIVABLE_ACCOUNT_LIST *
    receivable_account_list_new(
       LIST *journal_system_list() );

/* Process */
RECEIVABLE_ACCOUNT_LIST *
    receivable_account_list_calloc(
       void );

LIST *list = list_new();
```

```
for JOURNAL *journal in journal_system_list
{
    RECEIVABLE_ACCOUNT *
       receivable_account =
          receivable_account_getset(
             list,
             journal->account_name );

    if ( journal->debit_amount )
    {
       receivable_account->debit_amount +=
          journal->debit_amount;
    }
    else
    {
       receivable_account->debit_amount -=
          journal->credit_amount;
    }
}
```

```
/* Attribute */
LIST *list;
```



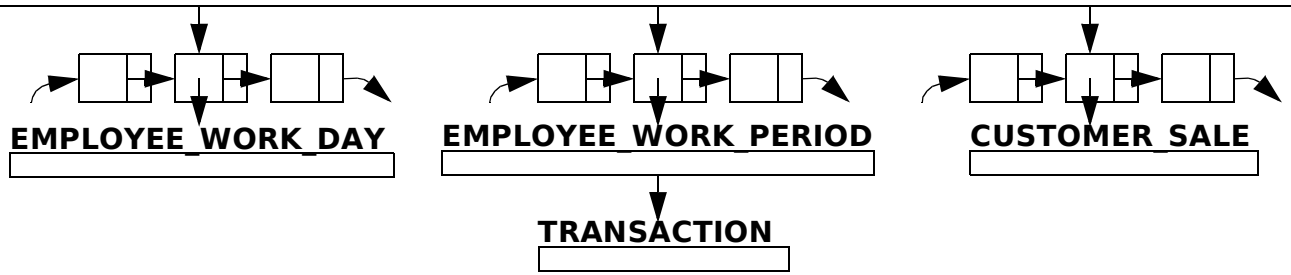
RECEIVABLE_ACCOUNT

```
/* Usage */
RECEIVABLE_ACCOUNT *receivable_account_getset(
    LIST *list,
    char *account_name );

/* Process */
for RECEIVABLE_ACCOUNT *receivable_account in list
{
    if ( strcmp(
       account_name,
       receivable_account->account_name ) == 0 )
    {
       return receivable_account;
    }
}

RECEIVABLE_ACCOUNT *receivable_account =
    receivable_account_new(
       account_name );

list_set( list, receivable_account );
```

```
/* Usage */
RECEIVABLE_ACCOUNT *receivable_account_new(
    char *account_name );

/* Process */
RECEIVABLE_ACCOUNT *receivable_account_calloc(
    void );

/* Attributes */
char *account_name;
double debit_amount;
```

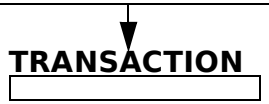
EMPLOYEE

```
/* Attributes */
char *full_name;
char *street_address;
double hourly_wage;
double period_salary;
double commission_sum_percent;
double gross_pay_year_to_date;
double net_pay_year_to_date;
char *federal_marital_status;
int federal_withholding_allowances;
int federal_withholding_additional_period_amount;
boolean state_withholding_exempt;
char *state_marital_status;
int state_withholding_allowances;
int state_itemized_deduction_allowances;
int retirement_contribution_plan_employee_period_amount;
int retirement_contribution_plan_employer_period_amount;
int health_insurance_employee_period_amount;
int health_insurance_employer_period_amount;
double union_dues_period_amount;
char *terminated_date;
LIST *employee_work_day_list;
LIST *employee_work_period_list;
LIST *customer_sale_list;
```



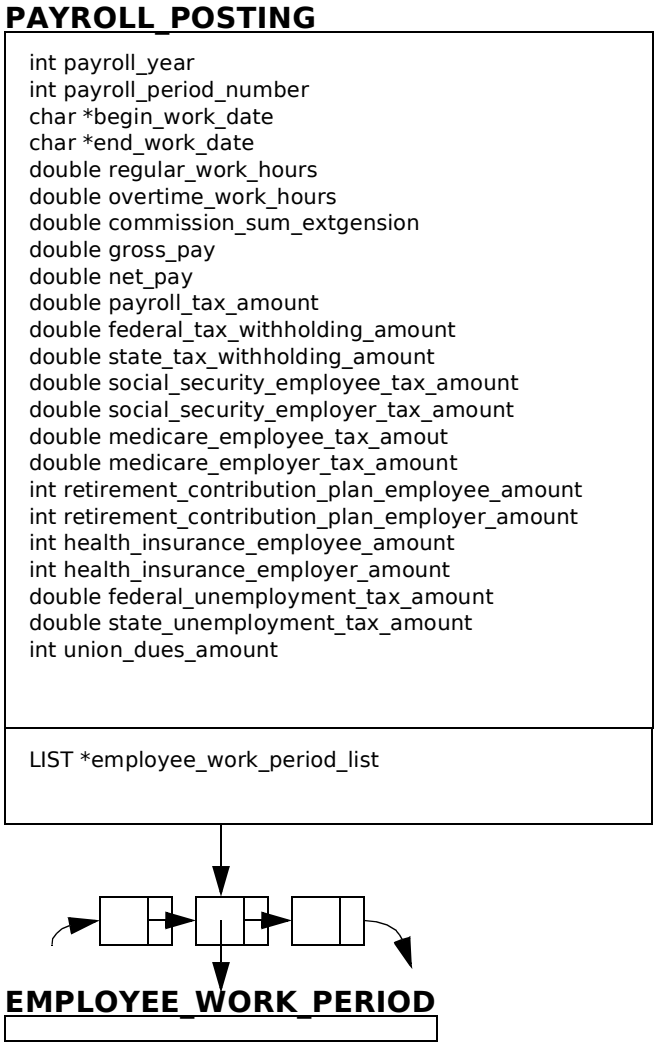
EMPLOYEE_WORK_PERIOD

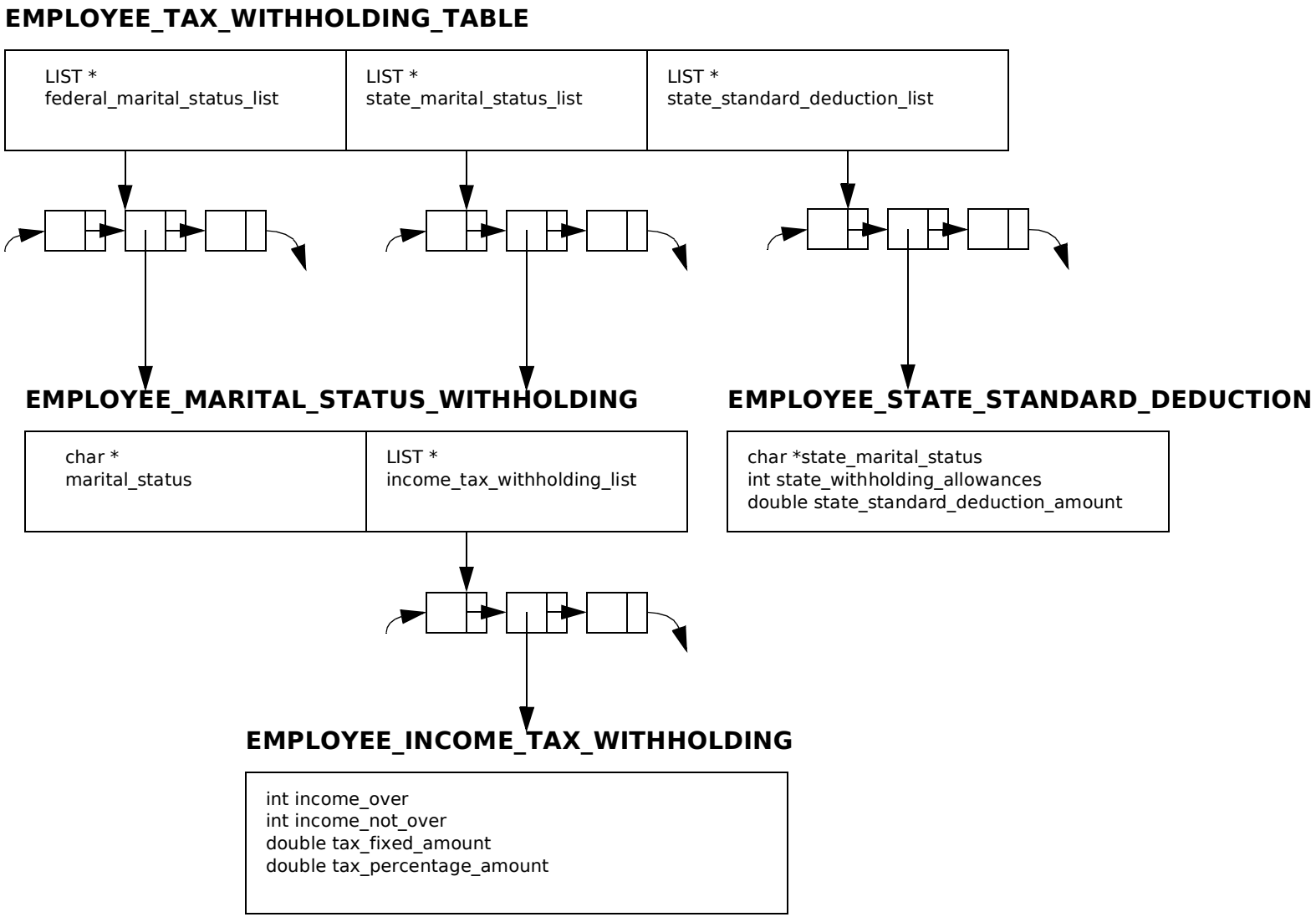
```
/* Attributes */
char *full_name;
char *street_address;
int payroll_year;
int payroll_period_number;
char *begin_work_date;
char *end_work_date;
double regular_work_hours;
double overtime_work_hours;
double commission_sum;
double gross_pay;
double net_pay;
double payroll_tax_amount;
double federal_tax_withholding_amount;
double state_tax_withholding_amount;
double social_security_employee_tax_amount;
double social_security_employer_tax_amount;
double medicare_employee_tax_amout;
double medicare_employer_tax_amount;
int retirement_contribution_plan_employee_amount;
int retirement_contribution_plan_employer_amount;
int health_insurance_employee_amount;
int health_insurance_employer_amount;
double federal_unemployment_tax_amount;
double state_unemployment_tax_amount;
int union_dues_amount;
char *transaction_date_time;
TRANSACTION *transaction;
```



EMPLOYEE_WORK_DAY

```
/* Attributes */
char *begin_work_date_time
char *end_work_date_time
double employee_work_hours
boolean overtime_work_day
```


PAYROLL_POSTING
int payroll_year
int payroll_period_number
char *begin_work_date
char *end_work_date
double regular_work_hours
double overtime_work_hours
double commission_sum_extgension
double gross_pay
double net_pay
double payroll_tax_amount
double federal_tax_withholding_amount
double state_tax_withholding_amount
double social_security_employee_tax_amount
double social_security_employer_tax_amount
double medicare_employee_tax_amout
double medicare_employer_tax_amount
int retirement_contribution_plan_employee_amount
int retirement_contribution_plan_employer_amount
int health_insurance_employee_amount
int health_insurance_employer_amount
double federal_unemployment_tax_amount
double state_unemployment_tax_amount
int union_dues_amount
LIST *employee_work_period_list
EMPLOYEE_WORK_PERIOD


EMPLOYEE_TAX_WITHHOLDING_TABLE
LIST *
federal_marital_status_list
LIST *
state_marital_status_list
LIST *
state_standard_deduction_list
EMPLOYEE_MARITAL_STATUS_WITHHOLDING
EMPLOYEE_STATE_STANDARD_DEDUCTION
char *
marital_status
LIST *
income_tax_withholding_list
char *state_marital_status
int state_withholding_allowances
double state_standard_deduction_amount
EMPLOYEE_INCOME_TAX_WITHHOLDING
int income_over
int income_not_over
double tax_fixed_amount
double tax_percentage_amount

FEEDER (1)

```
/* Usage */
FEEDER *feeder_fetch(
    char *application_name,
    char *login_name,
    char *feeder_account_name,
    char *exchange_format_filename,
    LIST *exchange_journal_list,
    double exchange_balance_amount,
    char *exchange_minimum_date_string );

/* Process */
FEEDER *feeder_calloc(
    void );

FEEDER_ACCOUNT *feeder_account_fetch(
    FEEDER_ACCOUNT_TABLE,
    feeder_account_name );

LIST *feeder_load_row_list(
    exchange_journal_list );

void feeder_load_row_calculate_balance_set(
    exchange_balance_amount,
    feeder_load_row_list() /* in/out */ );

char *account_uncleared_checks_string(
    ACCOUNT_UNCLEARED_CHECKS_KEY,
    __FUNCTION__ );

LIST *feeder_phrase_list(
    FEEDER_PHRASE_SELECT,
    FEEDER_PHRASE_TABLE );

int feeder_match_days_ago(
    const char *FEEDER_LOAD_TRANSACTION_DAYS_AGO,
    const int FEEDER_MATCH_DEFAULT_DAYS_AGO );

char *feeder_match_minimum_date(
    char *exchange_minimum_date_string,
    int feeder_match_days_ago() );

LIST *feeder_exist_row_list(
    FEEDER_ROW_TABLE,
    feeder_account_name,
    feeder_match_minimum_date );

LIST *feeder_matched_journal_list(
    feeder_account_name,
    FEEDER_ROW_TABLE,
    feeder_match_minimum_date(),
    account_uncleared_checks_string() );

LIST *feeder_row_list(
    feeder_account_name,
    feeder_account_fetch()->
      financial_institution_full_name,
    feeder_account_fetch()->
      financial_institution_street_address,
    account_uncleared_checks_string(),
    feeder_phrase_list(),
    feeder_exist_row_list(),
    feeder_matched_journal_list(),
    feeder_load_row_list() ) );

int feeder_row_count(
    feeder_row_list() );

if ( !feeder_row_count() ) return this;

int feeder_row_insert_count(
    LIST *feeder_row_list );

FEEDER_ROW *feeder_row_first_out_balance(
    feeder_row_list );

boolean account_accumulate_debit_boolean(
    feeder_account_name );

double feeder_prior_account_end_balance(
    FEEDER_LOAD_EVENT_TABLE,
    feeder_account_name,
    account_accumulate_debit_boolean() );

feeder_row_calculate_balance_set(
    feeder_row_list() /* in/out */,
    feeder_row_first_out_balance(),
    feeder_prior_account_end_balance() );

void feeder_row_status_set(
    feeder_row_list() /* in/out */
    feeder_row_first_out_balance() );

char *feeder_load_date_time =
    char *date_now19(
      date_utc_offset() );

char *feeder_row_account_end_date(
    feeder_row_list(),
    feeder_row_first_out_balance() );

if ( feeder_row_account_end_date() )
{
    double feeder_row_account_end_balance(
      feeder_row_list(),
      feeder_row_first_out_balance() );

    FEEDER_LOAD_EVENT *feeder_load_event =
      feeder_load_event_new(
        feeder_account_name,
        feeder_load_date_time,
        login_name,
        exchange_format_filename,
        feeder_row_account_end_date(),
        feeder_row_account_end_balance() );
}

char *feeder_parameter_end_balance_error(
    exchange_balance_amount,
    feeder_row_first_out_balance(),
    feeder_row_account_end_date(),
    feeder_row_account_end_balance() );
```

FEEDER (2)

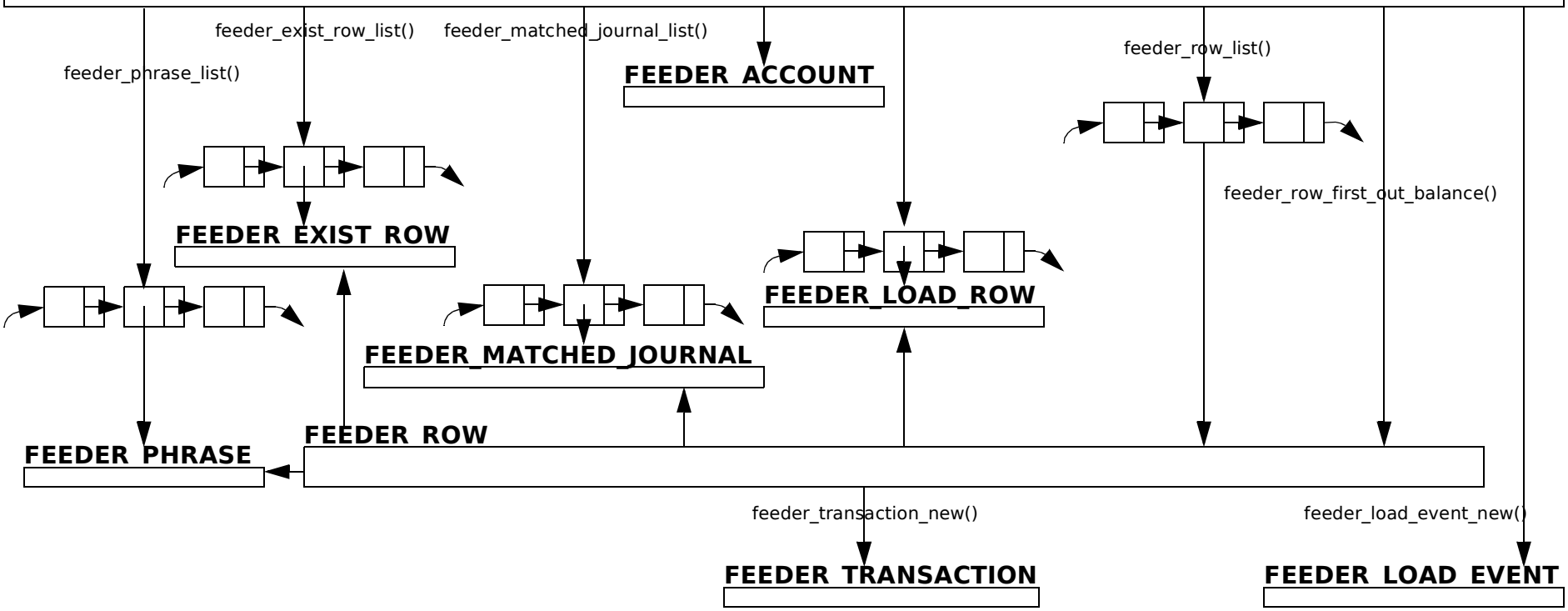
```
/* Usage */
double feeder_prior_account_end_balance(
    const char *FEEDER_LOAD_EVENT_TABLE,
    char *feeder_account_name,
    boolean account_accumulate_debit_boolean );

/* Process */
FEEDER_LOAD_EVENT *feeder_load_event_latest_fetch(
    FEEDER_LOAD_EVENT_TABLE,
    feeder_account_name );

if ( feeder_load_event_latest_fetch() )
{
    double prior_account_end_balance =
        feeder_load_event_latest_fetch()->
            feeder_row_account_end_balance;
}
else
{
    double prior_account_end_balance =
        journal_first_account_balance(
            feeder_account_name );

    if ( !account_accumulate_debit_boolean )
    {
        prior_account_end_balance =
            -prior_account_end_balance;
    }
}

/* Usage */
char *feeder_parameter_end_balance_error(
    double exchange_balance_amount,
    FEEDER_ROW *feeder->
        feeder_row_first_out_balance,
    char *feeder_row_account_end_date(),
    double feeder->
        feeder_row_account_end_balance );

/* Process */
```

FEEDER (3)

```
/* Usage */
void feeder_execute(
    char *process_name,
    FEEDER *feeder );

/* Process */
/* May reset */
/*  feeder_row-> */
/*     feeder_transaction-> */
/*     transaction-> */
/*     transaction_date_time */
/* and */
/* feeder_row-> */
/*     transaction_date_time */
feeder_row_transaction_insert(
    feeder->feeder_row_list,
    feeder->feeder_row_first_out_balance );

feeder_row_check_journal_update(
    feeder->feeder_row_list,
    feeder->feeder_row_first_out_balance );

feeder_row_journal_propagate(
    feeder->feeder_account_name,
    feeder->
       feeder_load_event->
       feeder_load_date_time,
    feeder->account_uncleared_checks_string );

feeder_load_event_insert(
    FEEDER_LOAD_EVENT_TABLE,
    FEEDER_LOAD_EVENT_INSERT,
    feeder->
       feeder_load_event->
       feeder_account_name,
    feeder->
       feeder_load_event->
       feeder_load_date_time,
    feeder->
       feeder_load_event->
       login_name,
    feeder->
       feeder_load_event->
       exchange_format_filename,
    feeder->
       feeder_load_event->
       feeder_row_account_end_date,
    feeder->
       feeder_load_event->
       feeder_row_account_end_balance );

feeder_row_list_insert(
    feeder->feeder_account_name,
    feeder->
       feeder_load_event->
       feeder_load_date_time,
    feeder->feeder_row_list,
    feeder->feeder_row_first_out_balance );

printf(
    "%s\n",
    feeder->parameter_end_balance_error );

printf(
    "<h3>Execute feeder row count: %d.</h3>\n",
    feeder->feeder_row_count );

process_execution_count_increment(
    process_name );

/* Usage */
void feeder_display(
    FEEDER *feeder );

/* Process */
if ( feeder->feeder_row_first_out_balance )
{
    printf( "<h1>Feeder Error Table</h1>\n" );

    feeder_row_error_display(
       feeder->feeder_row_list );
}

printf( "<h1>Transaction Table</h1>\n" );

feeder_row_list_display(
    feeder->feeder_row_list );

printf(
    "%s\n",
    feeder->parameter_end_balance_error );

printf(
    "<h3>Non-execute feeder row count: %d</h3>\n",
    feeder->feeder_row_count );

/* Usage */
boolean feeder_execute_boolean(
    boolean execute_boolean,
    FEEDER_ROW *feeder_row_first_out_balance );

/* Process */
if ( feeder_row_first_out_balance )
    return 0;
else
    return execute_boolean;
```

FEEDER (4)

```
/* Attributes */
char *feeder_account_name;
FEEDER_ACCOUNT *feeder_account;
LIST *feeder_load_row_list;
char *account_uncleared_checks_string;
LIST *feeder_phrase_list;
LIST *feeder_exist_row_list;
int match_days_ago;
char *match_minimum_date;
LIST *feeder_matched_journal_list;
LIST *feeder_row_list;
FEEDER_ROW *feeder_row_first_out_balance;
int feeder_row_count;
int feeder_row_insert_count;
boolean account_accumulate_debit_boolean;
double prior_account_end_balance;
char *feeder_row_account_end_date;
double feeder_row_account_end_balance;
char *feeder_load_date_time;
FEEDER_LOAD_EVENT *feeder_load_event;
char *parameter_end_balance_error;
```



FEEDER (5)

```
/* Driver */
EXCHANGE *exchange_parse(
   application_name,
   filename /* exchange_format_filename */,
   appaserver_parameter_upload_directory() );

if ( exchange_parse() )
{
   FEEDER *feeder_fetch(
      application_name,
      login_name,
      feeder_account_name,
      filename /* exchange_format_filename */,
      exchange->exchange_journal_list,
      exchange->balance_amount,
      exchange->minimum_date_string );
}

if ( feeder_fetch() )
{
   if ( !feeder_fetch()->feeder_row_count )
   {
      printf(
         "<h3>No new feeder rows to process.</h3>\n" );
   }
   else
   if (   execute_boolean
   &&  feeder_fetch()->feeder_row_insert_count )
   {
      feeder_execute(
         process_name,
         feeder_fetch() );
   }
   else
   {
      feeder_display( feeder_fetch() );
   }
}

if (   !feeder_fetch()
||     !feeder_fetch()->feeder_row_insert_count
||     execute_boolean )
{
   FEEDER_AUDIT_FETCH *feeder_audit_fetch(
      application_name,
      login_name,
      feeder_account_name );

   if ( !feeder_audit_fetch()->feeder_load_event )
   {
      printf(
      "<h3>Warning: no feeder load events.</h3>\n" );
   }
   else
   if ( !feeder_audit_fetch()->feeder_row_final_number )   }
   {
      printf(
      "<h3>"
      "ERROR: feeder_row_final_number is empty."
      "</h3>\n" );
   }
   else
   if ( !feeder_audit_fetch()->journal_latest )
   {
      printf(
      "<h3>Warning: "
      "no journal entries exist for account=%s"
      "</h3>\n",
      feeder_account_name );
   }
   else
   if ( !feeder_audit_fetch()->html_table )
   {
      printf(
      "<h3>ERROR: html_table is empty./h3>\n );
   }
   else
   {
      html_table_output(
         feeder_audit_fetch()->html_table,
         HTML_TABLE_ROWS_BETWEEN_HEADING );
   }
```

FEEDER_LOAD_ROW (1)

```
/* Usage */
FEEDER_LOAD_ROW *feeder_load_row_new(
    double exchange_journal_amount,
    char *exchange_journal_description,
    char *transaction_date_time,
    double debit_amount,
    double credit_amount,
    double journal_balance );

/* Process */
FEEDER_LOAD_ROW *feeder_load_row_calloc(
    void );

char *feeder_load_row_international_date(
    transaction_date_time );

char *feeder_load_row_description_space_trim(
    exchange_journal_description );

int feeder_load_row_check_number(
    feeder_load_row_description_space_trim() );

char *feeder_load_row_description_embedded(
    journal_balance,
    feeder_load_row_description_space_trim(),
    feeder_load_row_check_number() );

/* Attributes */
double exchange_journal_amount;
char *transaction_date_time;
double debit_amount;
double credit_amount;
double journal_balance;
char *international_date;
int check_number;
char *description_space_trim;
char *description_embedded;

/* Set externally */
/* Saved as file_row_amount */
double calculate_balance;
```

FEEDER_LOAD_ROW (2)

```
/* Usage */
char *feeder_load_row_description_space_trim(
    char *exchange_journal_description );

/* Process */
static char *sed_trim_double_spaces(
    exchange_journal_description );

char *strdup( sed_trim_double_spaces() );

/* Usage */
char *feeder_load_row_description_embedded(
    double feeder_load_row->journal_balance,
    char *feeder_load_row->description_space_trim,
    int feeder_load_row->check_number );

/* Process */
if ( check_number )
{
    return description_space_trim;
}

char *feeder_load_row_trim_date(
    char *description_space_trim );

char *feeder_load_row_description_build(
    double journal_balance,
    char *feeder_load_row_trim_date() );

char *feeder_load_row_description_crop(
    const int FEEDER_DESCRIPTION_SIZE,
    char *feeder_load_row_description_build() );

/* Usage */
void feeder_load_row_calculate_balance_set(
    double exchange_balance_amount,
    LIST *feeder_load_row_list /* in/out */ );

/* Process */
if ( list_tail( feeder_load_row_list ) )
do {
    FEEDER_LOAD_ROW *feeder_load_row =
        list_get( feeder_load_row_list );

    feeder_load_row->calculate_balance =
        exchange_balance_amount;

    exchange_balance_amount -=
        feeder_load_row->
            exchange_journal_amount;

} while ( list_previous( feeder_load_row_list ) );

/* Usage */
LIST *feeder_load_row_list(
    LIST *exchange_journal_list );

/* Process */
for EXCHANGE_JOURNAL *exchange_journal in
    exchange_journal_list
{
    FEEDER_LOAD_ROW *feeder_load_row_new(
        exchange_journal->amount,
        exchange_journal->description,
        exchange_journal->
            journal->
            transaction_date_time,
        exchange_journal->
            journal->
            debit_amount,
        exchange_journal->
            journal->
            credit_amount,
        exchange_journal->
            journal->
            balance );

    list_set(
        list,
        feeder_load_row_new() );
}

/* Usage */
int feeder_load_row_check_number(
    char *feeder_load_row->description_space_trim );

/* Process */

/* CHECK #:1827 */
int feeder_load_row_pound_colon_number(
    char *description_space_trim );

if ( feeder_load_row_check_pound_colon_number() )
        return this;

/* #1827 */
int feeder_load_row_pound_number(
    char *description_space_trim );

if ( feeder_load_row_pound_number() )
        return this;

/* check 1827 or CHECK 1827 */
int feeder_load_row_check_text_number(
    char *description_space_trim );

if ( feeder_load_row_check_text_number() )
        return this;

/* Usage */
int feeder_load_row_position_check_number(
    char *description_space_trim,
    int position,
    int strlen_substr );

/* Process */

/* Usage */
char *feeder_load_row_raw_display(
    FEEDER_LOAD_ROW *feeder_load_row );

/* Process */
```

FEEDER_ROW (1)

```
/* Usage */
LIST *feeder_row_list(
    char *feeder_account_name,
    char *financial_institution_full_name,
    char *financial_institution_street_address,
    char *account_uncleared_checks_string(),
    LIST *feeder_phrase_list(),
    LIST *feeder_exist_row_list(),
    LIST *feeder_matched_journal_list(),
    LIST *feeder_load_row_list );

/* Process */
char *minimum_transaction_date_time = (char *)0;

for FEEDER_LOAD_ROW *feeder_load_row in
    feeder_load_row_list
{
    FEEDER_ROW *feeder_row_new(
        feeder_account_name,
        financial_institution_full_name,
        financial_institution_street_address,
        account_uncleared_checks_string,
        feeder_phrase_list,
        feeder_exist_row_list,
        feeder_matched_journal_list,
        feeder_load_row,
        minimum_transaction_date_time,
        ++feeder_row_number );

    if ( !feeder_row_new() ) continue;

    if (feeder_row_new()->transaction_date_time )
    {
        minimum_transaction_date_time =
            feeder_row_new()->
                transaction_date_time );
    }
    list_set(
        list,
        feeder_row_new() );
}

/* Usage */
FEEDER_ROW *feeder_row_new(
    char *feeder_account_name,
    char *financial_institution_full_name,
    char *financial_institution_street_address,
    char *account_uncleared_checks_string,
    LIST *feeder_phrase_list(),
    LIST *feeder_exist_row_list(),
    LIST *feeder_matched_journal_list(),
    FEEDER_LOAD_ROW *feeder_load_row,
    char *minimum_transaction_date_time,
    int feeder_row_number );

/* Process */
FEEDER_ROW *feeder_row_calloc(
    void );

enum feeder_row_status feeder_row_status =
    feeder_row_status_out_of_balance;

FEEDER_EXIST_ROW *feeder_exist_row_seek(
    feeder_load_row->international_date,
    feeder_load_row->description_embedded,
    feeder_exist_row_list );

if ( feeder_exist_row_seek() )
{
    return this;
}

FEEDER_MATCHED_JOURNAL *
    feeder_matched_journal =
        feeder_matched_journal_check_seek(
            feeder_account_name,
            account_uncleared_checks_string,
            feeder_load_row->check_number,
            feeder_load_row->exchange_journal_amount,
            feeder_matched_journal_list );

if ( !feeder_matched_journal )
{
    FEEDER_MATCHED_JOURNAL *
        feeder_matched_journal =
            feeder_matched_journal_amount_seek(
                feeder_load_row->exchange_journal_amount,
                feeder_matched_journal_list );
}

if ( feeder_matched_journal )
{
    feeder_matched_journal->taken = 1;
}

if ( !feeder_matched_journal )
{
    FEEDER_PHRASE *feeder_phrase_seek(
        financial_institution_full_name,
        financial_institution_street_address,
        feeder_load_row->description_space_trim,
        feeder_phrase_list );
}

if (    !feeder_matched_journal
&&  !feeder_phrase_seek() )
{
    return this;
}
```

FEEDER_ROW (2)

```
/* Process (continued) */
if ( feeder_phrase_seek() )
{
   char *transaction_date_time =
      feeder_row_transaction_date_time(
         char *feeder_load_row->international_date,
         char *minimum_transaction_date_time );

   FEEDER_TRANSACTION *
      feeder_transaction =
         feeder_transaction_new(
            feeder_account_name,
            feeder_phrase_seek,
            feeder_load_row->exchange_journal_amount,
            transaction_date_time,
            feeder_load_row->description_embedded
               /* memo */ );
}
else
/* Must be feeder_matched_journal */
{
    char *transaction_date_time =
       feeder_matched_journal->
          transaction_date_time;
}

/* Usage */
void feeder_row_calculate_balance_set(
   LIST *feeder_row_list(),
   FEEDER_ROW *feeder_row_first_out_balance(),
   double feeder_prior_account_end_balance() );

/* Process */
for FEEDER_ROW *feeder_row in feeder_row_list
{
   if ( feeder_row == feeder_row_first_out_balance )
         break;

   feeder_row->calculate_balance =
      feeder_row_calculate_balance(
         feeder_row,
         feeder_prior_account_end_balance );

   feeder_prior_account_end_balance =
      feeder_row->calculate_balance;
}

/* Usage */
double feeder_row_calculate_balance(
   FEEDER_ROW *feeder_row,
   double feeder_prior_account_end_balance() );

/* Process */
if ( feeder_row->feeder_exist_row_seek )
{
   double calculate_balance =
      feeder_row->
         feeder_exist_row_seek->
         file_row_balance;
}
else
{
   double calculate_balance =
      feeder_prior_account_end_balance +
      feeder_row->
         feeder_load_row->
         exchange_journal_amount;
}

/* Usage */
FEEDER_ROW *feeder_row_first_out_balance(
   LIST *feeder_row_list );

/* Process */
for FEEDER_ROW *feeder_row in feeder_row_list
{
   if (   !feeder_row->feeder_exist_row_seek
   &&  !feeder_row->feeder_matched_journal
   &&  !feeder_row->feeder_phrase_seek )
         return feeder_row;
}
return NULL;

/* Usage */
void feeder_row_status_set(
   LIST *feeder_row_list() /* in/out */,
   FEEDER_ROW *feeder_row_first_out_balance() );

/* Process */
for FEEDER_ROW *feeder_row in feeder_row_list
{
   if ( feeder_row == feeder_row_first_out_balance )
         break;

   feeder_row->feeder_row_status =
      feeder_row_status_evaluate(
         feeder_row->
            feeder_load_row->
            calculate_balance,
         feeder_row->calculate_balance );
}

/* Usage */
enum feeder_row_status feeder_row_status_evaluate(
   double feeder_load_row_calculate_balance,
   double feeder_row_calculate_balance );

/* Process */
boolean float_money_virtually_same(
   feeder_load_row_calculate_balance,
   feeder_row_calculate_balance );

if ( float_money_virtually_same() )
   return feeder_row_status_okay;
else
   return feeder_row_status_out_of_balance;
```

FEEDER_ROW (3)

```
/* Usage */
char *feeder_row_system_string(
    char *FEEDER_ROW_SELECT,
    char *FEEDER_ROW_TABLE,
    char *where );

/* Process */

/* Usage */
LIST *feeder_row_system_list(
    feeder_row_system_string() );

/* Process */
FILE *feeder_row_input_pipe(
    char *feeder_row_system_string );

for char input[] in
    string_input(
       feeder_row_input_pipe() )
{
    list_set(
       list,
       FEEDER_ROW *
          feeder_row_parse(
             input ) );
}

pclose( feeder_row_input_pipe() );

/* Usage */
FEEDER_ROW *feeder_row_parse(
   char *input );

/* Process */
FEEDER_ROW *feeder_row_calloc();

/* Usage */
int feeder_row_final_number(
    char *FEEDER_ROW_TABLE,
    char *feeder_account_name,
    char *feeder_load_event->
       feeder_load_date_time );;

/* Process */
char *feeder_load_event_primary_where(
    feeder_account_name,
    feeder_load_date_time );
```

```
char *feeder_row_final_number_select(
    void );

sprintf(
    char system_string[],
    "select.sh \"%s\" %s \"%s\"",
    feeder_row_final_number_select(),
    FEEDER_ROW_TABLE,
    feeder_load_event_primary_where() );

int string_atoi( string_pipe_fetch( system_string[] ) );

/* Usage */
int feeder_row_maximum_transaction_date_time(
    char *FEEDER_ROW_TABLE,
    char *feeder_account_name,
    char *feeder_load_date_time );

/* Process */
char *feeder_load_event_primary_where(
    feeder_account_name,
    feeder_load_date_time );

void sprintf(
    char system_string[],
    "select.sh \"%s\" %s \"%s\"",
    "max( transaction_date_time )",
    FEEDER_ROW_TABLE,
    feeder_load_event_primary_where() );

char *string_pipe_fetch( system_string[] );

/* Usage */
int feeder_row_minimum_transaction_date_time(
    char *FEEDER_ROW_TABLE,
    char *feeder_account_name,
    char *feeder_load_date_time );

/* Process */
char *feeder_load_event_primary_where(
    feeder_account_name,
    feeder_load_date_time );

void sprintf(
    char system_string[],
    "select.sh \"%s\" %s \"%s\"",
    "min( transaction_date_time )",
    FEEDER_ROW_TABLE,
    feeder_load_event_primary_where() );

char *string_pipe_fetch( system_string[] );

/* Usage */
FEEDER_ROW *feeder_row_fetch(
    char *feeder_account_name,
    char *feeder_load_event->feeder_load_date_time,
    int feeder_row_number );

/* Process */
char *feeder_row_primary_where(
   char *feeder_account,
   char *feeder_load_date_time,
   int feeder_row_number );

char *feeder_row_system_string(
    FEEDER_ROW_SELECT,
    FEEDER_ROW_TABLE,
    feeder_row_primary_where() );

FEEDER_ROW *feeder_row_parse(
   string_pipe_fetch(
      feeder_row_system_string() );

/* Usage */
double feeder_row_account_end_balance(
   LIST *feeder_row_list,
   FEEDER_ROW *feeder_row_first_out_balance() );

/* Process */
for FEEDER_ROW *feeder_row in feeder_row_list
{
   if ( feeder_row == feeder_row_first_out_balance )
         break;

   account_end_balance =
      feeder_row->calculate_balance;
}

/* Usage */
char *feeder_row_account_end_date(
   LIST *feeder_row_list,
   FEEDER_ROW *feeder_row_first_out_balance() );

/* Process */
```

FEEDER_ROW (4)

```
/* Usage */
char *feeder_row_transaction_date_time(
    char *international_date,
    char *minimum_transaction_date_time );

/* Process */
strcpy(
    transaction_date_time[],
    transaction_increment_date_time(
        international_date
            /* transaction_date_string */ ) );

if (    minimum_transaction_date_time
&&  strcmp(
            transaction_date_time,
            minimum_transaction_date_time ) < 0 )
{
    DATE *date_19new(
        minimum_transaction_date_time );

    date_increment_second(
        date_19_new(),
        1 );

    char *date_display19(
        date_19_new() );

    strcpy(
        transaction_date_time[],
        date_display19() );
}

/* Usage */
int feeder_row_count(
    LIST *feeder_row_list );

/* Process */
int list_length( feeder_row_list );

/* Usage */
int feeder_row_insert_count(
    LIST *feeder_row_list );

/* Process */
for FEEDER_ROW *feeder_row in feeder_row_list
{
    if (   feeder_row->feeder_matched_journal
    ||     feeder_row->feeder_phrase_seek )
    {
        insert_count++;
    }
}

/* Usage */
void feeder_row_error_display(
    LIST *feeder_row_list );

/* Process */
LIST *feeder_row_error_extract_list(
    LIST *feeder_row_list );

feeder_row_list_display(
    feeder_row_error_extract_list()
        /* feeder_row_list */ );

/* Usage */
void feeder_row_list_display(
    LIST *feeder_row_list );

/* Process */
char *system_string =
    char *feeder_row_list_display_system_string(
        void );

FILE *display_pipe =
    FILE *feeder_row_list_display_pipe(
        system_string );

if ( !list_length( feeder_row_list ) )
{
    pclose( display_pipe );
    return;
}

for FEEDER_ROW *feeder_row in feeder_row_list
{
    if ( !display_pipe )
    {
        FILE *display_pipe =
            FILE *feeder_row_list_display_pipe(
                system_string );
    }

    FILE *display_pipe =
        FILE *feeder_row_display_output(
            display_pipe,
            feeder_row );
}

if ( display_pipe ) pclose( display_pipe );

fflush( stdout );

/* Usage */
FILE *feeder_row_display_output(
    FILE *display_pipe,
    FEEDER_ROW *feeder_row );

/* Process */
char *feeder_row_display_results(
    FEEDER_EXIST_ROW *
        feeder_exist_row_seek,
    FEEDER_MATCHED_JOURNAL *
        feeder_matched_journal,
    FEEDER_PHRASE *
        feeder_phrase_seek );

char *feeder_row_status_string(
    feeder_row->feeder_row_status );

if (    feeder_row->feeder_transaction
&&  feeder_row->feeder_transaction->transaction )
{
    pclose( display_pipe );
    fflush( stdout );
    display_pipe = (FILE *)0;

    journal_list_html_display(
        feeder_row->
            feeder_transaction->
            transaction->
            journal_list,
        feeder_row->
            feeder_transaction->
            transaction->
            transaction_date_time,
        feeder_row->
            feeder_transaction->
            transaction->
            memo,
        (char *)0 /* transaction_full_name */ );
}
```

FEEDER_ROW (5)

```
/* Usage */
char *feeder_row_status_string(
    enum feeder_row_status feeder_row_status );

/* Process */

/* Usage */
void feeder_row_list_insert(
     char *feeder_account_name,
     char *feeder_load_event->feeder_load_date_time,
     LIST *feeder_row_list,
     FEEDER_ROW *feeder_row_first_out_balance );

/* Process */
char *feeder_row_list_insert_system_string(
     char *FEEDER_ROW_INSERT_COLUMNS,
     char *FEEDER_ROW_TABLE,
     char SQL_DELIMITER );

FILE *feeder_row_list_insert_open(
     feeder_row_list_insert_system_string() );

for FEEDER_ROW *feeder_row in feeder_row_list
{
     if (feeder_row == feeder_row_first_out_balance )
          break;

     if ( feeder_row->feeder_exist_row_seek )
          continue;

if (   !feeder_row->feeder_phrase_seek
&&  !feeder_row->feeder_matched_journal )
{
   continue;
}

JOURNAL *feeder_row_journal(
   FEEDER_MATCHED_JOURNAL *
     feeder_row->feeder_matched_journal,
   FEEDER_PHRASE *
     feeder_row->feeder_phrase_seek,
   /* See feeder_row_transaction_date_time_reset() */
   char *feeder_row->
     transaction_date_time );

char *feeder_row_phrase(
   FEEDER_PHRASE *
     feeder_row->feeder_phrase_seek );

void feeder_row_insert(
   FILE *feeder_row_list_insert_open(),
   char *feeder_account_name,
   char *feeder_load_date_time,
   char *feeder_row->
     feeder_load_row->
     international_date,
     int feeder_row->
       feeder_row_number,
     char *feeder_row_journal()->
       full_name,
     char *feeder_row_journal()->
       street_address,
     char *feeder_row_journal()->
       transaction_date_time,
     char *feeder_row->
       feeder_load_row->
       description_embedded,
     double feeder_row->
       feeder_load_row->
       exchange_journal_amount,
     double feeder_row->
       feeder_load_row->
       calculate_balance,
     double feeder_row->
       calculate_balance,
     int feeder_row->
       feeder_load_row->
       check_number,
     char *feeder_row_phrase(),
     char SQL_DELIMITER );
}
pclose( feeder_row_list_insert_open() );
```

FEEDER_ROW (6)

```
/* Usage */
/* May reset */
/* feeder_row->*/
/*     feeder_transaction->*/
/*     transaction_date_time */
/* and */
/* feeder_row-> */
/*     transaction_date_time */
void feeder_row_transaction_insert(
    LIST *feeder->feeder_row_list,
    FEEDER_ROW *feeder->
       feeder_row_first_out_balance );

/* Process */
LIST *feeder_row_extract_transaction_list(
    feeder_row_list,
    feeder_row_first_out_balance ),

/* May reset transaction->transaction_date_time */
void transaction_list_insert(
    feeder_row_extract_transaction_list(),
    0 /* not insert_journal_list_boolean */,
    0 /* not transaction_lock_boolean */ );

void transaction_journal_list_insert(
    feeder_row_extract_transaction_list(),
    1 /* with_propagate */ );

/* Set each feeder_row->transaction_date_time */
void feeder_row_transaction_date_time_set(
    LIST *feeder_row_list /* in/out */ );

/* Usage */
void feeder_row_journal_propagate(
    char *feeder_account_name,
    char *feeder->
       feeder_load_event->
       feeder_load_date_time,
    char *feeder->account_uncleared_checks_string );

/* Process */
char *feeder_row_minimum_transaction_date_time(
    FEEDER_ROW_TABLE,
    feeder_account_name,
    feeder_load_date_time );

JOURNAL_PROPAGATE *journal_propagate =
    journal_propagate_new(
       feeder_row_minimum_transaction_date_time(),
       feeder_account_name );

if ( journal_propagate )
{
    journal_list_update(
       journal_propagate->
          update_statement_list );
}

JOURNAL_PROPAGATE *journal_propagate =
    journal_propagate_new(
       feeder_row_minimum_transaction_date_time(),
       account_uncleared_checks_string );

if ( journal_propagate )
{
    journal_list_update(
       journal_propagate->
          update_statement_list );
}

/* Usage */
LIST *feeder_row_extract_transaction_list(
    LIST *feeder_row_list,
    FEEDER_ROW *feeder_row_first_out_balance );

/* Process */
for FEEDER_ROW *feeder_row in
    feeder_row_list
{
    if (feeder_row == feeder_row_first_out_balance )
          break;

    if ( feeder_row->feeder_transaction )
    {
       list_set(
          transaction_list,
          feeder_row->feeder_transaction );
    }
}

/* Usage */
void feeder_row_check_journal_update(
    LIST *feeder->feeder_row_list,
    FEEDER_ROW *feeder->
       feeder_row_first_out_balance );

/* Process */
FILE *feeder_row_check_journal_update_pipe(
    void );

for FEEDER_ROW *feeder_row in feeder_row_list
{
    if ( feeder_row == feeder_row_first_out_balance )
          break;

    if (   feeder_row->feeder_matched_journal
    &&  feeder_row->
             feeder_matched_journal->
             check_update_statement )
    {
       fprintf(
          feeder_row_check_journal_update_pipe(),
          "%s\n",
          feeder_row->
             feeder_matched_journal->
             check_update_statement );
    }
}
pclose( feeder_row_check_journal_update_pipe() );

/* Usage */
char *feeder_row_raw_display(
    FEEDER_ROW *feeder_row );

/* Process */
```

FEEDER_ROW (7)

```
/* Attributes */
char *feeder_account_name;
FEEDER_LOAD_ROW *feeder_load_row;
int feeder_row_number;
FEEDER_EXIST_ROW *feeder_exist_row_seek;
FEEDER_MATCHED_JOURNAL *feeder_matched_journal;
FEEDER_PHRASE *feeder_phrase_seek;
FEEDER_TRANSACTION *feeder_transaction;
char *transaction_date_time;

/* Set by feeder_row_parse() */
char *feeder_load_date_time;
char *feeder_date;
char *full_name;
char *street_address;
char *file_row_description;
double file_row_amount;
double file_row_balance;
int check_number;
char *feeder_phrase;

/* Set externally */
double calculate_balance;
enum feeder_row_status feeder_row_status;
```

enum feeder_row_status

- feeder_row_status_okay
- feeder_row_status_out_of_balance
- feeder_row_status_cannot_determine

feeder_exist_row_seek()

FEEDER_EXIST_ROW

feeder_matched_journal_check_seek()
or
feeder_matched_journal_amount_seek()

FEEDER_MATCHED_JOURNAL

feeder_phrase_seek()

FEEDER_PHRASE

FEEDER_LOAD_ROW

feeder_transaction_new()

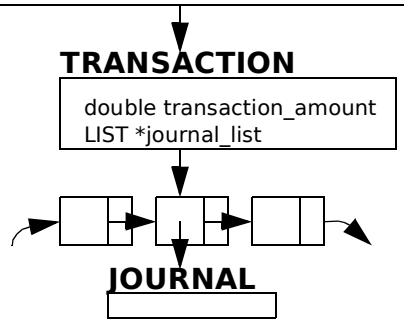
FEEDER_TRANSACTION

FEEDER_EXIST_ROW

```
/* Usage */
LIST *feeder_exist_row_list(
    const char *FEEDER_ROW_TABLE,
    char *feeder_account_name,
    char *feeder_match_minimum_date );

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

char *feeder_exist_row_where(
    char *feeder_account_name,
    char *feeder_match_minimum_date );

char *feeder_exist_row_system_string(
    const char *FEEDER_EXIST_ROW_SELECT,
    const char *FEEDER_ROW_TABLE,
    char *feeder_exist_row_where() );

FILE *feeder_exist_row_input_open(
    char *feeder_exist_row_system_string() );

while ( input[] = string_input( input_open ) )
{
    list_set(
        list,
        FEEDER_EXIST_ROW *
            feeder_exist_row_parse(
                input ) );
}
pclose( feeder_exist_row_input_open() );

/* Usage */
FEEDER_EXIST_ROW *
    feeder_exist_row_parse(
        char *input );

/* Process */
FEEDER_EXIST_ROW *feeder_exist_row_calloc(
    void );

/* Usage */
FEEDER_EXIST_ROW *
    feeder_exist_row_seek(
        char *international_date,
        char *description_embedded,
        LIST *feeder_exist_row_list );

/* Process */
for FEEDER_EXIST_ROW *feeder_exist_row in
    feeder_exist_row_list
{
    if (   strcmp(  feeder_exist_row->feeder_date,
                    international_date ) == 0
    &&  strcmp(  feeder_exist_row->file_row_description,
                    description_embedded ) == 0 )
    {
        return this;
    }
}
return NULL;

/* Attributes */
char *feeder_date;
char *file_row_description;
char *transaction_date_time;
double file_row_amount;
double file_row_balance;
```

FEEDER_TRANSACTION

```
/* Usage */
FEEDER_TRANSACTION *
    feeder_transaction_new(
       char *feeder_account_name,
       FEEDER_PHRASE *feeder_phrase_seek(),
       double exchange_journal_amount,
       char *transaction_date_time,
       char *memo );

/* Process */
FEEDER_TRANSATION *feeder_transaction_calloc(
    void );

if ( exchange_journal_amount < 0.0 )
{
    char *debit_account_name =
          feeder_phrase_seek->
             nominal_account;

    char *credit_account_name = feeder_account_name;

    exchange_journal_amount =
       -exchange_journal_amount;
}
else
{
    char *debit_account_name = feeder_account_name;

    char *credit_account_name =
          feeder_phrase_seek->
             nominal_account;
}

TRANSACTION *transaction =
    transaction_binary(
       feeder_phrase_seek->full_name,
       feeder_phrase_seek->street_address,
       transaction_date_time,
       exchange_journal_amount
          /* transaction_amount */,
       memo,
       debit_account_name,
       credit_account_name );

/* Attributes */
char *debit_account;
char *credit_account;
TRANSACTION *transaction;
```



FEEDER_MATCHED_JOURNAL

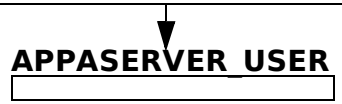
```
/* Usage */
LIST *feeder_matched_journal_list(
    char *feeder_account_name,
    char *FEEDER_ROW_TABLE,
    char *feeder_match_minimum_date(),
    char *account_uncleared_checks_string() );

/* Process */
char *feeder_matched_journal_subquery(
    char *feeder_account_name,
    char *account_uncleared_checks_string,
    char *JOURNAL_TABLE,
    char *FEEDER_ROW_TABLE );

char *feeder_matched_journal_where(
    char *feeder_account_name,
    char *account_uncleared_checks_string,
    char *feeder_matched_journal_subquery(),
    char *feeder_match_minimum_date );

LIST *journal_system_list(
    char *journal_system_string(
       JOURNAL_SELECT,
       JOURNAL_TABLE,
       feeder_matched_journal_list_where() ),
    0 /* not fetch_account */,
    0 /* not fetch_subclassification */,
    0 /* not fetch_element */,
    1 /* fetch_transaction */ );

if ( !list_rewind( journal_system_list() ) return NULL;

LIST *list = list_new();

for JOURNAL *journal in journal_system_list()
{
    list_set(
       list,
       FEEDER_MATCHED_JOURNAL *
          feeder_matched_journal_new(
             journal ) );
}

/* Usage */
FEEDER_MATCHED_JOURNAL *
    feeder_matched_journal_new(
       JOURNAL *journal );

/* Process */
FEEDER_MATCHED_JOURNAL *
    feeder_matched_journal_calloc(
       void );

double feeder_matched_journal_amount(
    double journal->debit_amount,
    double journal->credit_amount );

/* Usage */
FEEDER_MATCHED_JOURNAL *
    feeder_matched_journal_check_seek(
       char *feeder_account_name,
       char *account_uncleared_checks_string,
       int check_number,
       double exchange_journal_amount,
       LIST *feeder_matched_journal_list() );

/* Process */
for FEEDER_MATCHED_JOURNAL *
    feeder_matched_journal
       in feeder_matched_journal_list
{
    if ( feeder_matched_journal->taken ) continue;

    if (   feeder_matched_journal->check_number ==
           check_number
    &&  float_money_virtually_same(
              feeder_matched_journal->amount,
              exchange_journal_amount ) )
    {
       char *
       feeder_matched_journal_check_update_statement(
          char *JOURNAL_TABLE,
          char *feeder_account_name,
          char *account_uncleared_checks_string,
          char *feeder_matched_journal->full_name,
          char *feeder_matched_journal->street_address,
          char *feeder_matched_journal->
             transaction_date_time );

       return this;
    }
}
return NULL;

/* Usage */
FEEDER_MATCHED_JOURNAL *
    feeder_matched_journal_amount_seek(
       double exchange_journal_amount,
       LIST *feeder_matched_journal_list() );

/* Process */
for FEEDER_MATCHED_JOURNAL *
    feeder_matched_journal in
       feeder_matched_journal_list
{
    if ( feeder_matched_journal->taken )
          continue;

    if ( feeder_matched_journal->check_number )
          continue;

    if (float_money_virtually_same(
          feeder_matched_journal->amount,
          exchange_journal_amount ) )
    {
       return this;
    }
}
return NULL;

/* Attributes */
char *full_name;
char *street_address;
char *transaction_date_time;
char *account_name;
double debit_amount;
double credit_amount;
double amount;

/* Set externally */
int check_number;
boolean taken;
char *check_update_statement;
```

FEEDER_PHRASE

```
/* Usage */
LIST *feeder_phrase_list(
    char *FEEDER_PHRASE_SELECT,
    char *FEEDER_PHRASE_TABLE );

/* Process */
char *appaserver_system_string(
    char *FEEDER_PHRASE_SELECT,
    char *FEEDER_PHRASE_TABLE,
    (char *)0 /* where */ );

FILE *appaserver_input_pipe(
    char *appaserver_system_string() );

LIST *list = list_new();

while ( input[] =
    string_input(
       appaserver_input_pipe() ) )
{
    list_set(
       list,
       feeder_phrase_parse( input ) );
}

pclose( appaserver_input_pipe() );

/* Usage */
FEEDER_PHRASE *feeder_phrase_parse(
    char *input );

/* Process */
FEEDER_PHRASE *feeder_phrase_new(
    phrase );

/* Usage */
FEEDER_PHRASE *feeder_phrase_new(
    char *phrase );

/* Process */
FEEDER_PHRASE *feeder_phrase_calloc( void );

/* Usage */
FEEDER_PHRASE *feeder_phrase_seek(
    char *financial_institution_full_name,
    char *financial_institution_street_address,
    char *feeder_load_row->description_space_trim,
    LIST *feeder_phrase_list() );

/* Process */
char feeder_component[];

for FEEDER_PHRASE *feeder_phrase in
    feeder_phrase_list
{
    for ( int piece_number = 0;
          piece(
             feeder_component,
             FEEDER_PHRASE_DELIMITER,
             feeder_phrase->phrase,
             piece_number );
          piece_number++ )
    {
       if ( string_exists_substr(
             description_space_trim /* string */,
             feeder_component /* substring */ ) )
       {
          FEEDER_PHRASE *feeder_phrase_entity_set(
             financial_institution_full_name,
             financial_institution_street_address,
             feeder_phrase );
       }
    }
}
return NULL;

/* Usage */
static char *feeder_phrase_primary_where(
    char *feeder_phrase );

/* Process */

/* Usage */
FEEDER_PHRASE *feeder_phrase_entity_set(
    char *financial_institution_full_name,
    char *financial_institution_street_address,
    FEEDER_PHRASE *feeder_phrase );

/* Process */
char *feeder_phrase_full_name(
    char *financial_institution_full_name,
    char *feeder_phrase->full_name );

char *feeder_phrase_street_address(
    char *financial_institution_street_address,
    feeder_phrase->street_address );

/* Attributes */
char *phrase;
char *nominal_account;
char *full_name;
char *street_address;
```

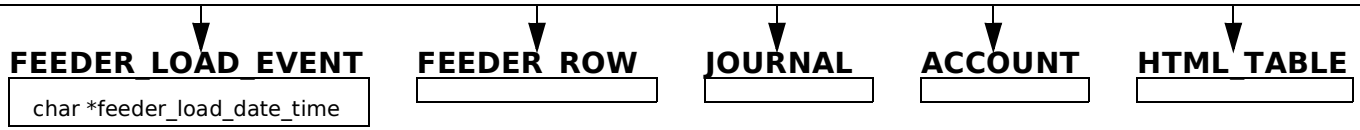
FEEDER_LOAD_EVENT

```
/* Usage */
FEEDER_LOAD_EVENT *feeder_load_event_new(
    char *feeder_account_name,
    char *feeder_load_date_time,
    char *login_name,
    char *exchange_format_filename,
    char *feeder_row_account_end_date(),
    double feeder_row_account_end_balance() );

/* Process */
FEEDER_LOAD_EVENT *feeder_load_event_calloc(
    void );

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

/* Usage */
FEEDER_LOAD_EVENT *feeder_load_event_fetch(
    char *feeder_account_name,
    char *feeder_load_date_time );

/* Process */
char *feeder_load_event_primary_where(
    char *feeder_account_name,
    char *feeder_load_date_time );

char *feeder_load_event_system_string(
    char *FEEDER_LOAD_EVENT_SELECT,
    char *FEEDER_LOAD_EVENT_TABLE,
    char *feeder_load_event_primary_where() );

FILE *feeder_load_event_input_open(
    char *feeder_load_event_system_string() );

char *string_input(
    input[],
    feeder_load_event_input_open() );

FEEDER_LOAD_EVENT *feeder_load_event_parse(
    input );

pclose( feeder_load_event_input_open() );

/* Usage */
FEEDER_LOAD_EVENT *feeder_load_event_parse(
    char *input );

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

/* Usage */
FEEDER_LOAD_EVENT *
    feeder_load_event_latest_fetch(
      const char *FEEDER_LOAD_EVENT_TABLE,
      char *feeder_account_name );

/* Process */
char *feeder_load_event_account_where(
    char *feeder_account_name );

char *feeder_load_event_latest_system_string(
    const char *FEEDER_LOAD_EVENT_TABLE,
    char *feeder_load_event_account_where() );

char *feeder_load_event_latest_date_time(
    char *feeder_load_event_latest_system_string() );

FEEDER_LOAD_EVENT *
    feeder_load_event_fetch(
      feeder_account_name,
      feeder_load_event_latest_date_time() );

/* Usage */
void feeder_load_event_insert(
    const char *FEEDER_LOAD_EVENT_TABLE,
    const char *FEEDER_LOAD_EVENT_INSERT,
    char *feeder_load_event->feeder_account_name,
    char *feeder_load_event->feeder_load_date_time,
    char *feeder_load_event->
      appaserver_user->
      full_name,
    char *feeder_load_event->
      appaserver_user->
      street_address,
    char *feeder_load_event->
      exchange_format_filename,
    char *feeder_load_event->
      feeder_row_account_end_date,
    double feeder_load_event->
      feeder_row_account_end_balance );

/* Process */
char *feeder_load_event_insert_system_string(
    const char *FEEDER_LOAD_EVENT_TABLE,
    const char *FEEDER_LOAD_EVENT_INSERT,
    const char SQL_DELIMITER );

FILE *feeder_load_event_insert_open(
    char *feeder_load_event_insert_system_string() );

void feeder_load_event_insert_pipe(
    FILE *feeder_load_event_insert_open(),
    const char SQL_DELIMITER,
    char *feeder_account_name,
    char *feeder_load_date_time,
    char *full_name,
    char *street_address,
    char *exchange_format_filename,
    char *feeder_row_account_end_date,
    double feeder_row_account_end_balance );

pclose( feeder_load_event_insert_open() );

/* Attributes */
char *feeder_account;
char *feeder_load_date_time;
char *exchange_format_filename;
char *feeder_row_account_end_date;
double feeder_row_account_end_balance;
APPASERVER_USER *appaserver_user;
```



FEEDER_AUDIT (1)

```
/* Usage */
FEEDER_AUDIT *feeder_audit_fetch(
    char *application_name,
    char *login_name,
    char *feeder_account_name );

/* Process */
FEEDER_AUDIT *feeder_audit_calloc(
    void );

FEEDER_LOAD_EVENT *feeder_load_event =
    feeder_load_event_latest_fetch(
        FEEDER_LOAD_EVENT_TABLE,
        feeder_account_name );

if ( !feeder_load_event ) return this;

int feeder_row_final_number(
    FEEDER_ROW_TABLE,
    feeder_account_name,
    feeder_load_event->feeder_load_date_time );

if ( !feeder_row_final_number() ) return this;

FEEDER_ROW *feeder_row_fetch(
    feeder_account_name,
    feeder_load_event->feeder_load_date_time,
    feeder_row_final_number()
        /* feeder_row_number */ );

char *feeder_audit_end_transaction_date_time(
    const char *TRANSACTION_DATE_CLOSE_TIME,
    char *feeder_row_fetch()->feeder_date );

JOURNAL *journal_latest(
    JOURNAL_TABLE,
    feeder_account_name,
    feeder_audit_end_transaction_date_time(),
    0 /* not fetch_transaction_boolean */,
    1 /* zero_balance_boolean */ );

if ( !journal_latest() ) return this;

ACCOUNT *account_fetch(
    feeder_account_name,
    1 /* fetch_subclassification */,
    1 /* fetch element */ );

double feeder_audit_credit_balance_negate(
    double journal_latest()->balance
        /* journal_balance */,
    boolean account_fetch()->
        subclassification->
        element->
        accumulate_debit );

double feeder_audit_balance_difference(
    double feeder_row_fetch->calculate_balance,
    double feeder_audit_credit_balance_negate()
        /* journal_balance */ );

boolean feeder_audit_difference_zero(
    double feeder_audit_balance_difference() );

char *feeder_audit_account_display(
    char *feeder_account_name );

HTML_TABLE *html_table =
    html_table_new(
        FEEDER_AUDIT_HTML_TITLE,
        feeder_audit_account_display() /* sub_title */,
        (char *)0 /* sub_sub_title */ );

html_table->column_list =
    LIST *feeder_audit_html_column_list( void );

html_table->row_list = list_new();

list_set(
    html_table->row_list,
    HTML_ROW *feeder_audit_html_row(
        application_name,
        login_name,
        feeder_row_fetch()
            /* feeder_row */,
        journal_latest(),
        feeder_audit_credit_balance_negate()
            /* journal_balance */,
        feeder_audit_balance_difference(),
        feeder_audit_difference_zero() ) );

/* Attributes */
FEEDER_LOAD_EVENT *feeder_load_event;
int feeder_row_final_number;
FEEDER_ROW *feeder_row_fetch;
char *end_transaction_date_time;
JOURNAL *journal_latest;
ACCOUNT *account_fetch;
double credit_balance_negate;
double balance_difference;
boolean difference_zero;
HTML_TABLE *html_table;
```



FEEDER_AUDIT (2)

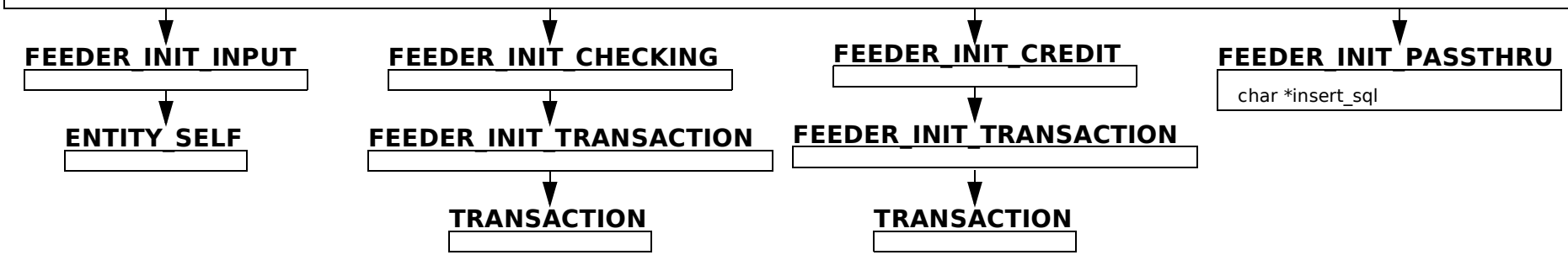
```
/* Usage */
HTML_ROW *feeder_audit_html_row(
    char *application_name,
    char *login_name,
    FEEDER_ROW *feeder_row,
    JOURNAL *journal_latest(),
    double journal_balance,
    double feeder_audit_balance_difference(),
    boolean feeder_audit_difference_zero() );

/* Process */
date_convert_format_enum
    date_convert_login_name_enum(
       application_name,
       login_name );

char *date_convert_return_date_string(
    date_convert_international
       /* source_enum */,
    date_convert_login_name_enum()
       /* destination_enum */,
    feeder_row->feeder_date
       /* source_date_string */ );

LIST *feeder_audit_html_cell_list(
    int feeder_row->feeder_row_number,
    char *feeder_row->full_name,
    char *feeder_row->file_row_description,
    char *feeder_row->transaction_date_time,
    char *date_convert_return_date_string()
       /* feeder_date */,
    double feeder_row->file_row_amount,
    double feeder_row->calculate_balance,
    char *journal_latest->full_name,
    char *journal_latest->transaction_date_time,
    double journal_balance,
    double feeder_audit_balance_difference,
    boolean feeder_audit_difference_zero );

/* Driver */
void html_table_output(
    feeder_audit->html_table,
    HTML_TABLE_ROWS_BETWEEN_HEADING );
```

FEEDER_ACCOUNT

```
/* Usage */
FEEDER_ACCOUNT *feeder_account_fetch(
    const char *FEEDER_ACCOUNT_TABLE,
    char *feeder_account_name );

/* Process */
static char *feeder_account_primary_where(
    FEEDER_ACCOUNT_PRIMARY_KEY,
    feeder_account_name );

char *appaserver_system_string(
    FEEDER_ACCOUNT_SELECT,
    FEEDER_ACCOUNT_TABLE,
    feeder_account_primary_where() );

char *string_pipe_fetch(
    appaserver_system_string() );

if ( !string_pipe_fetch() ) exit( 1 );

FEEDER_ACCOUNT *feeder_account_parse(
    feeder_account_name,
    string_pipe_fetch() );

/* Usage */
FEEDER_ACCOUNT *feeder_account_parse(
    char *feeder_account_name,
    char *string_pipe_fetch() );

/* Process */

/* Usage */
FEEDER_ACCOUNT *feeder_account_new(
    char *feeder_account_name );

/* Process */
FEEDER_ACCOUNT *feeder_account_calloc(
    void );

/* Usage */
char *feeder_account_primary_where(
    const char *FEEDER_ACCOUNT_PRIMARY_KEY,
    char *feeder_account_name );

/* Process */
static char *security_sql_injection_escape(
    SECURITY_ESCAPE_CHARACTER_STRING,
    feeder_account_name );

snprintf(
    static primary_where[],
    sizeof ( primary_where ),
    "%s = '%s'",
    FEEDER_ACCOUNT_PRIMARY_KEY,
    security_sql_injection_escape() );

free( security_sql_injection_escape() );

/* Attributes */
char *feeder_account_name;
char *financial_institution_full_name;
char *financial_institution_street_address;
```

FEEDER_INIT (1)

```
/* Usage */
FEEDER_INIT *feeder_init_new(
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *financial_institution_full_name,
    char *financial_institution_street_address,
    boolean checking_boolean,
    double exchange_journal_begin_amount,
    char *exchange_minimum_date_string );

/* Process */
FEEDER_INIT *feeder_init_calloc(
    void );

FEEDER_INIT_INPUT *feeder_init_input_new(
    application_name,
    financial_institution_full_name,
    checking_boolean,
    exchange_minimum_date_string );

if ( feeder_init_input_new()->
       institution_missing_boolean )
          return this;

if ( feeder_init_input_new()->account_exist_boolean )
       return this;

if ( !feeder_init_input_new()->entity_self )
       return this;

if ( checking_boolean )
{
    FEEDER_INIT_CHECKING *feeder_init_checking_new(
       0 /* not execute_boolean */,
       exchange_journal_begin_amount,
       exchange_minimum_date_string,
       feeder_init_input->account_name,
       feeder_init_input->entity_self->full_name,
       feeder_init_input->entity_self->street_address );
}
else
{
    FEEDER_INIT_CREDIT *feeder_init_credit_new(
       0 /* not execute_boolean */,
       -exchange_journal_begin_amount
          /* negate_exchange_journal_begin_amount */,
       exchange_minimum_date_string,
       feeder_init_input->account_name,
       feeder_init_input->entity_self->full_name,
       feeder_init_input->entity_self->street_address );
}

FEEDER_INIT_PASSTHRU *feeder_init_passthru_new(
    checking_boolean,
    feeder_init_input->entity_self->full_name,
    feeder_init_input->entity_self->street_address );

char *feeder_init_account_insert_sql(
    ACCOUNT_TABLE,
    SUBCLASSIFICATION_CASH,
    SUBCLASSIFICATION_CURRENT_LIABILITY,
    checking_boolean,
    feeder_init_input->account_name );

char *feeder_init_feeder_account_insert_sql(
    FEEDER_ACCOUNT_TABLE,
    FEEDER_ACCOUNT_PRIMARY_KEY,
    feeder_init_input->account_name,
    financial_institution_full_name,
    financial_institution_street_address );

LIST *feeder_init_insert_sql_list(
    char *feeder_init_passthru_new()->insert_sql,
    char *feeder_init_account_insert_sql(),
    char *feeder_init_feeder_account_insert_sql() );

static char *feeder_init_trial_balance_system_string(
    const char *FEEDER_INIT_TRIAL_EXECUTABLE,
    char *session_key,
    char *login_name,
    char *role_name );

static char *feeder_init_activity_system_string(
    const char *FEEDER_INIT_ACTIVITY_EXECUTABLE,
    char *session_key,
    char *login_name,
    char *role_name );

static char *feeder_init_position_system_string(
    const char *FEEDER_INIT_POSITION_EXECUTABLE,
    char *session_key,
    char *login_name,
    char *role_name );

/* Attributes */
FEEDER_INIT_INPUT *feeder_init_input;
FEEDER_INIT_CHECKING *feeder_init_checking;
FEEDER_INIT_CREDIT *feeder_init_credit;
FEEDER_INIT_PASSTHRU *feeder_init_passthru;
char *account_insert_sql;
char *feeder_account_insert_sql;
LIST *insert_sql_list;
char *trial_balance_system_string;
char *activity_system_string;
char *position_system_string;
```



FEEDER_INIT (2)

```
/* Usage */
void feeder_init_transaction_html_display(
    FEEDER_INIT_CHECKING *feeder_init_checking,
    FEEDER_INIT_CREDIT *feeder_init_credit );

/* Process */
if ( feeder_init_checking )
{
    transaction_html_display(
        feeder_init_checking->
            feeder_init_transaction->
            transaction );
}
else
if ( feeder_init_credit )
{
    transaction_html_display(
        feeder_init_credit->
            feeder_init_transaction->
            transaction );
}

/* Usage */
char *feeder_init_account_insert_sql(
    const char *ACCOUNT_TABLE,
    const char *ACCOUNT_PRIMARY_KEY
    const char *SUBCLASSIFICATION_CASH,
    const char *SUBCLASSIFICATION_CURRENT_LIABILITY,
    const char *ACCOUNT_CREDIT_CARD_KEY,
    boolean checking_boolean,
    char *feeder_init_input->account_name );

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

list_set(
    insert_datum_list,
    insert_datum_new(
        ACCOUNT_PRIMARY_KEY /* attribute_name */,
        account_name /* datum */,
        1 /* primary_key_index */,
        0 /* not attribute_is_number */ ) );

const char *feeder_init_subclassification(
    const char *SUBCLASSIFICATION_CASH,
    const char *SUBCLASSIFICATION_CURRENT_LIABILITY,
    boolean checking_boolean );

list_set(
    insert_datum_list,
    insert_datum_new(
        "subclassification" /* attribute_name */,
        feeder_init_subclassification() /* datum */,
        0 /* primary_key_index */,
        0 /* not attribute_is_number */ ) );

if ( !checking_boolean )
{
    list_set(
        insert_datum_list,
        insert_datum_new(
            "hard_coded_account_key",
            ACCOUNT_CREDIT_CARD_KEY,
            0 /* primary_key_index */,
            0 /* not attribute_is_number */ ) );
}

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(
    ACCOUNT_TABLE,
    insert_datum_attribute_name_list_string(),
    insert_datum_value_list_string() );

/* Usage */
char *feeder_init_feeder_account_insert_sql(
    const char *FEEDER_ACCOUNT_TABLE,
    const char *FEEDER_ACCOUNT_PRIMARY_KEY,
    char *feeder_init_input->account_name,
    char *financial_institution_full_name,
    char *financial_institution_street_address );

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

list_set(
    insert_datum_list,
    insert_datum_new(
        FEEDER_ACCOUNT_PRIMARY_KEY
            /* attribute_name */,
        account_name /* datum */,
        1 /* primary_key_index */,
        0 /* not attribute_is_number */ ) );

list_set(
    insert_datum_list,
    insert_datum_new(
        "full_name" /* attribute_name */,
        financial_institution_full_name /* datum */,
        0 /* primary_key_index */,
        0 /* not attribute_is_number */ ) );

list_set(
    insert_datum_list,
    insert_datum_new(
        "street_address" /* attribute_name */,
        financial_institution_full_name /* datum */,
        0 /* primary_key_index */,
        0 /* not attribute_is_number */ ) );

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(
    FEEDER_ACCOUNT_TABLE,
    insert_datum_attribute_name_list_string(),
    insert_datum_value_list_string() );
```

FEEDER_INIT (3)

```
/* Driver */
if ( feeder_init->
       feeder_init_input->
       institution_missing_boolean )
{
   printf(
      "%s\n",
      FEEDER_INIT_INSTITUTION_MISSING_MESSAGE );
   exit( 0 );
}

if ( feeder_init->
       feeder_init_input->
       account_exist_boolean )
{
   printf(
      "%s\n",
      FEEDER_INIT_ACCOUNT_EXIST_MESSAGE );
   exit( 0 );
}

if ( !feeder_init->
       feeder_init_input->
       entity_self )
{
   printf(
      "%s\n",
      FEEDER_INIT_ENTITY_SELF_MESSAGE );
   exit( 0 );
}

if ( feeder_init->
       feeder_init_passthru->
       exist_boolean )
{
   printf(
      "%s\n",
      FEEDER_INIT_PASSTHRU_EXIST_MESSAGE );
}

if ( !execute_boolean )
{
   if ( feeder_init->
          feeder_init_input->
          date_recent_boolean )
   {
      printf(
         "%s\n",
         FEEDER_INIT_RECENT_MESSAGE );
   }

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

   void feeder_init_transaction_html_display(
      feeder_init->feeder_init_checking,
      feeder_init->feeder_init_credit );

   printf(
      "%s\n",
      IMPORT_PREDICT_SHORTCUT_MESSAGE );
}
```

FEEDER_INIT (4)

```
/* Driver (continued) */
if ( execute_boolean )
{
   char *error_string =
      sql_execute(
         SQL_EXECUTABLE,
         feeder_init->
            feeder_init_input->
            appaserver_error_filespecification,
         feeder_init->insert_sql_list,
         (char *)0 /* sql_statement */ );

   if ( error_string )
   {
      appaserver_error_message_file(
         application_name,
         login_name,
         error_string );
   }

   if ( checking_boolean )
   {
      FEEDER_INIT_CHECKING *
         feeder_init_checking_new(
            1 /* execute_boolean */,
            exchange_journal_begin_amount,
            exchange_minimum_date_string,
            feeder_init_input->account_name,
            feeder_init_input->entity_self->full_name,
            feeder_init_input->
               entity_self->
               street_address );
   }
   else
   {
      FEEDER_INIT_CREDIT *
         feeder_init_credit_new(
            1 /* execute_boolean */,
            -exchange_journal_begin_amount
            /* negate_exchange_journal_begin_amount */,
            exchange_minimum_date_string,
            feeder_init_input->account_name,
            feeder_init_input->entity_self->full_name,
            feeder_init_input->
               entity_self->
               street_address );
   }

   if ( feeder_init->feeder_init_checking )
   {
      void feeder_init_transaction_insert(
         feeder_init->
            feeder_init_checking->
            feeder_init_transaction );
   }
   else
   {
      void feeder_init_transaction_insert(
         feeder_init->
            feeder_init_credit->
            feeder_init_transaction );
   }

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


   void feeder_init_transaction_html_display(
      feeder_init->feeder_init_checking,
      feeder_init->feeder_init_credit );

   printf( "<h3>Reminder</h3>\n" );

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

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

   security_system(
      SECURITY_FORK_CHARACTER,
      SECURITY_FORK_STRING,
      feeder_init->trial_balance_system_string );

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

   security_system(
      SECURITY_FORK_CHARACTER,
      SECURITY_FORK_STRING,
      feeder_init->activity_system_string );

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

   security_system(
      SECURITY_FORK_CHARACTER,
      SECURITY_FORK_STRING,
      feeder_init->position_system_string );

   printf(
      "%s\n",
      FEEDER_INIT_UPLOAD_MESSAGE );
}

printf(
   "%s\jn",
   FEEDER_INIT_MESSAGES_AVAILABLE_MESSAGE );
```

FEEDER_INIT_INPUT

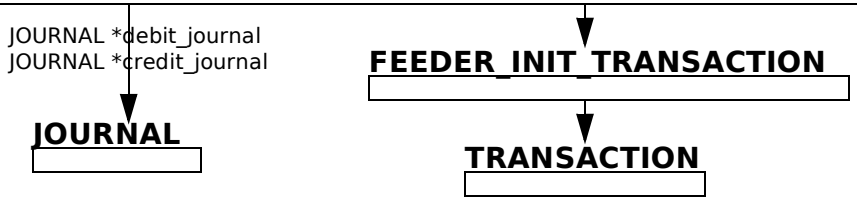
```
/* Usage */
FEEDER_INIT_INPUT *feeder_init_input_new(
    char *application_name;
    char *financial_institution_full_name,
    boolean checking_boolean,
    char *exchange_minimum_date_string );

/* Process */
FEEDER_INIT_INPUT *feeder_init_input_calloc(
    void );

boolean feeder_init_input_institution_missing_boolean(
    const char *SECURITY_FORBID_CHARACTER_STRING,
    char *financial_institution_full_name );

if ( feeder_init_input_institution_missing_boolean() )
       return this;

char *feeder_init_input_account_name(
    financial_institution_full_name,
    checking_boolean );

boolean feeder_init_input_account_exist_boolean(
    const char *ACCOUNT_TABLE,
    char *feeder_init_input_account_name() );

if ( feeder_init_input_account_exist_boolean() )
       return this;

char *date_now_yyyy_mm_dd( date_utc_offset() );

boolean feeder_init_input_date_recent_boolean(
    const int FEEDER_INIT_INPUT_DAYS_AGO,
    char *exchange_minimum_date_string,
    char *date_now_yyyy_mm_dd() );

ENTITY_SELF *entity_self =
    entity_self_fetch(
       0 /* not fetch_entity_boolean */ );

char *appaserver_error_filespecification(
    application_name );

/* Usage */
char *feeder_init_input_account_name(
    char *financial_institution_full_name,
    boolean checking_boolean );

/* Process */
static char *string_mnemonic(
    financial_institution_full_name );

if ( checking_boolean )
{
    snprintf(
       static account_name[],
       sizeof ( account_name ),
       "%s_checking",
       string_mnemonic() );
}
else
{
    snprintf(
       static account_name[],
       sizeof ( account_name ),
       "%s_creditcard",
       string_mnemonic() );
}

/* Attributes */
boolean institution_missing_boolean;
char *account_name;
boolean account_exist_boolean;
char *date_now_yyyy_mm_dd;
boolean date_recent_boolean;
ENTITY_SELF *entity_self;
char *appaserver_error_filespecification;
```

FEEDER_INIT_CHECKING

```
/* Usage */
FEEDER_INIT_CHECKING *feeder_init_checking_new(
    boolean execute_boolean,
    double exchange_journal_begin_amount,
    char *exchange_minimum_date_string,
    char *feeder_init_input->account_name,
    char *feeder_init_input->entity_self_full_name,
    char *feeder_init_input->entity_self_street_address );

/* Process */
FEEDER_INIT_CHECKING *feeder_init_checking_calloc(
    void );

char *feeder_init_transaction_equity_account_name(
    ACCOUNT_EQUITY_KEY );
```

```
JOURNAL *debit_journal =
    JOURNAL *feeder_init_transaction_journal(
        exchange_journal_begin_amount,
        account_name,
        1 /* debit_boolean */,
        execute_boolean /* fetch_account_boolean */ );

JOURNAL *credit_journal =
    JOURNAL *feeder_init_transaction_journal(
        exchange_journal_begin_amount,
        feeder_init_transaction_equity_account_name(),
        0 /* not debit_boolean */,
        execute_boolean /* fetch_account_boolean */ );
```

```
FEEDER_INIT_TRANSACTION *
    feeder_init_transaction_new(
        TRANSACTION_BEGIN_TIME,
        exchange_journal_begin_amount,
        exchange_minimum_date_string,
        entity_self_full_name,
        entity_self_street_address,
        debit_journal,
        credit_journal );

/* Attributes */
char *feeder_init_transaction_equity_account_name;
JOURNAL *debit_journal;
JOURNAL *credit_journal;
FEEDER_INIT_TRANSACTION *feeder_init_transaction;
```

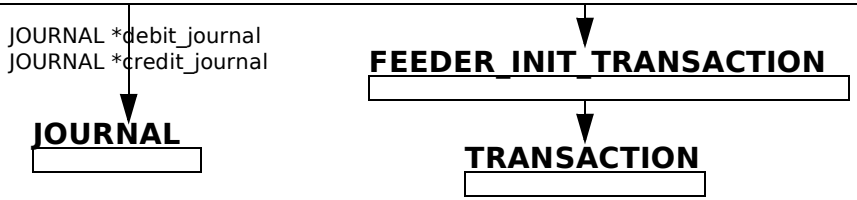


FEEDER_INIT_CREDIT

```
/* Usage */
FEEDER_INIT_CREDIT *feeder_init_credit_new(
    boolean execute_boolean,
    double negate_exchange_journal_begin_amount,
    char *exchange_minimum_date_string,
    char *feeder_init_input->account_name,
    char *feeder_init_input->entity_self_full_name,
    char *feeder_init_input->entity_self_street_address );

/* Process */
FEEDER_INIT_CREDIT *feeder_init_credit_calloc(
    void );

char *feeder_init_transaction_equity_account_name(
    ACCOUNT_EQUITY_KEY );
```

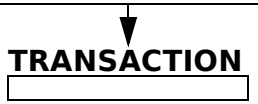
```
JOURNAL *debit_journal =
    JOURNAL *feeder_init_transaction_journal(
        negate_exchange_journal_begin_amount,
        feeder_init_transaction_equity_account_name(),
        1 /* debit_boolean */,
        execute_boolean /* fetch_account_boolean */ );

JOURNAL *credit_journal =
    JOURNAL *feeder_init_transaction_journal(
        negate_exchange_journal_begin_amount,
        account_name,
        0 /* not debit_boolean */,
        execute_boolean /* fetch_account_boolean */ );
```

```
FEEDER_INIT_TRANSACTION *
    feeder_init_transaction_new(
        TRANSACTION_BEGIN_TIME,
        negate_exchange_journal_begin_amount,
        exchange_minimum_date_string,
        entity_self_full_name,
        entity_self_street_address,
        debit_journal,
        credit_journal );

/* Attributes */
char *feeder_init_transaction_equity_account_name;
JOURNAL *debit_journal;
JOURNAL *credit_journal;
FEEDER_INIT_TRANSACTION *feeder_init_transaction;
```



FEEDER_INIT_TRANSACTION

```
/* Usage */
FEEDER_INIT_TRANSACTION *
    feeder_init_transaction_new(
        const char *TRANSACTION_BEGIN_TIME,
        double exchange_journal_begin_amount,
        char *exchange_minimum_date_string,
        char *entity_self_full_name,
        char *entity_self_street_address,
        JOURNAL *debit_journal,
        JOURNAL *credit_journal );

/* Process */
FEEDER_INIT_TRANSACTION *
    feeder_init_transaction_calloc(
        void );

char *feeder_init_transaction_date_time(
    const char *TRANSACTION_BEGIN_TIME,
    char *exchange_minimum_date_string );

TRANSACTION *transaction =
    transaction_new(
        entity_self_full_name,
        entity_self_street_address,
        feeder_init_transaction_date_time() );

transaction->transaction_amount =
    float_abs( exchange_journal_begin_amount );

transaction->memo =
    TRANSACTION_OPEN_MEMO;

transaction->journal_list = list_new();

list_set(
    transaction->journal_list,
    debit_journal );

list_set(
    transaction->journal_list,
    credit_journal );

/* Usage */
char *feeder_init_transaction_equity_account_name(
    const char *ACCOUNT_EQUITY_KEY );

/* Process */
char *account_hard_coded_account_name(
    ACCOUNT_EQUITY_KEY,
    0 /* not warning_only */,
    __FUNCTION__ /* calling_function_name */ );

/* Usage */
JOURNAL *feeder_init_transaction_journal(
    double exchange_journal_begin_amount,
    char *account_name,
    boolean debit_boolean,
    boolean fetch_account_boolean );

/* Process */
JOURNAL *journal_new(
    (char *)0 /* full_name */,
    (char *)0 /* street_address */,
    (char *)0 /* transaction_date_time */,
    account_name );

if ( fetch_account_boolean )
{
    journal_new()->account =
        account_fetch(
            journal_new()->account_name,
            1 /* fetch_subclassification */,
            1 /* fetch_element */ );
}

if ( debit_boolean )
{
    journal_new()->debit_amount =
        exchange_journal_begin_amount;
}

else
{
    journal_new()->credit_amount =
        exchange_journal_begin_amount;
}

/* Usage */
void feeder_init_transaction_insert(
    FEEDER_INIT_TRANSACTION *
        feeder_init_transaction );

/* Process */
(void)transaction_insert(
    feeder_init_transaction->
        transaction->
        full_name,
    feeder_init_transaction->
        transaction->
        street_address,
    feeder_init_transaction->
        transaction->
        transaction_date_time,
    feeder_init_transaction->
        transaction->
        transaction_amount,
    0 /* check_number */,
    feeder_init_transaction->
        transaction->
        memo,
    'n' /* lock_transaction_yn */,
    feeder_init_transaction->
        transaction->
        journal_list,
    1 /* insert_journal_list_boolean */ );

/* Attributes */
char *date_time;
TRANSACTION *transaction;
```

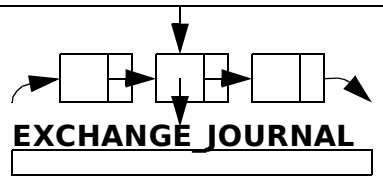


FEEDER_INIT_PASSTHRU

```
/* Usage */
FEEDER_INIT_PASSTHRU *feeder_init_passthru_new(
    boolean checking_boolean,
    char *entity_self_full_name,
    char *entity_self_street_address );

/* Process */
FEEDER_INIT_PASSTHRU *feeder_init_passthru_calloc(
    void );

char *feeder_init_passthru_account_name(
    ACCOUNT_PASSTHRU_KEY );

const char *feeder_init_passthru_feeder_phrase(
    const char *FEEDER_INIT_CASH_FEEDER_PHRASE,
    const char *FEEDER_INIT_CARD_FEEDER_PHRASE,
    boolean checking_boolean );

char *feeder_init_passthru_exist_system_string(
    FEEDER_PHRASE_TABLE,
    feeder_init_passthru_feeder_phrase() );

boolean feeder_init_passthru_exist_boolean(
    char *feeder_init_passthru_exist_system_string() );

if ( feeder_init_passthru_exists_boolean() ) return this;

char *feeder_init_passthru_insert_sql(
    FEEDER_PHRASE_TABLE,
    feeder_init_passthru_account_name(),
    entity_self_full_name,
    entity_self_street_address,
    feeder_init_passthru_feeder_phrase() );

/* Usage */
char *feeder_init_passthru_insert_sql(
    const char *FEEDER_PHRASE_TABLE,
    char *account_name,
    char *entity_self_full_name,
    char *entity_self_street_address,
    char *passthru_feeder_phrase );

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

list_set(
    insert_datum_list,
    insert_datum_new(
        "feeder_phrase" /* attribute_name */,
        passthru_feeder_phrase /* datum */,
        1 /* primary_key_index */,
        0 /* not attribute_is_number */ ) );

list_set(
    insert_datum_list,
    insert_datum_new(
        "nominal_account" /* attribute_name */,
        account_name /* datum */,
        0 /* primary_key_index */,
        0 /* not attribute_is_number */ ) );

list_set(
    insert_datum_list,
    insert_datum_new(
        "full_name" /* attribute_name */,
        entity_self_full_name /* datum */,
        0 /* primary_key_index */,
        0 /* not attribute_is_number */ ) );

list_set(
    insert_datum_list,
    insert_datum_new(
        "street_address" /* attribute_name */,
        entity_self_street_address /* datum */,
        0 /* primary_key_index */,
        0 /* not attribute_is_number */ ) );

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(
    FEEDER_PHRASE_TABLE,
    insert_datum_attribute_name_list_string(),
    insert_datum_value_list_string() );

/* Usage */
char *feeder_init_passthru_exist_system_string(
    const char *FEEDER_PHRASE_TABLE,
    char *passthru_feeder_phrase );

/* Process */
static char *feeder_phrase_primary_where(
    passthru_feeder_phrase );

snprintf(
    system_string[],
    sizeof ( system_string ),
    "select.sh \"count(1)\" %s \"%s\"",
    FEEDER_PHRASE_TABLE,
    feeder_phrase_primary_where() );

/* Usage */
char *feeder_init_passthru_account_name(
    const char *ACCOUNT_PASSTHRU_KEY );

/* Process */
char *account_hard_coded_account_name(
    ACCOUNT_PASSTHRU_KEY,
    0 /* not warning_only */,
    __FUNCTION__ /* calling_function_name */ );

/* Attributes */
char *account_name;
char *feeder_phrase;
char *exist_system_string;
boolean exist_boolean;
char *insert_sql;
```

EXCHANGE (1)

```
/* Usage */
EXCHANGE *exchange_parse(
    char *application_name,
    char *exchange_format_filename,
    char *appaserver_parameter->upload_directory );

/* Process */
EXCHANGE *exchange_calloc(
    void );

static char *exchange_filespecification(
    char *application_name,
    char *exchange_format_filename,
    char *upload_directory );

FILE *appaserver_input_file(
    exchange_filespecification() );

LIST *list_stream_fetch(
    appaserver_input_file() );

fclose( appaserver_input_file() );

boolean exchange_open_tag_boolean(
    EXCHANGE_OFX_TAG,
    list_stream_fetch() /* list */ );

if ( !exchange_open_tag_boolean() )
{
    return this;
}

char *exchange_financial_institution(
    EXCHANGE_ORG_TAG,
    list_stream_fetch() /* list */ );

LIST *exchange_journal_list = list_new();

(void)list_rewind( list_stream_fetch() );

while ( !list_past_end( list_stream_fetch() ) )
{
    EXCHANGE_JOURNAL *exchange_journal_extract(
        list_stream_fetch /* list */ );

    if ( exchange_journal_extract() )
    {
        list_set_order(
            exchange_journal_list,
            exchange_journal_extract(),
            exchange_journal_compare_function() );
    }
}

if ( !list_length( exchange_journal_list ) ) return this;

double exchange_balance_amount(
    EXCHANGE_BALAMT_TAG,
    list_stream_fetch() /* list */ );

double exchange_journal_begin_amount(
    exchange_journal_list /* in/out */,
    exchange_balance_amount() );

char *exchange_minimum_date_string(
    EXCHANGE_JOURNAL *
        list_first( exchange_journal_list )
            /* first_exchange_journal */ );

/* Attributes */
char *exchange_format_filename;
char *filespecification;
LIST *list_stream_fetch;
boolean open_tag_boolean;
char *financial_institution;
LIST *exchange_journal_list;
double balance_amount;
double exchange_journal_begin_amount;
char *minimum_date_string;
```



EXCHANGE (2)

```
/* Usage */
LIST *exchange_tag_seek_list(
    const char *tag,
    char *stop_tag,
    LIST *list );

/* Process */
while ( !list_past_end( list ) )
{
    char *list_get( list );

    if ( stop_tag )
    {
       boolean exchange_tag_boolean(
          (const char *)stop_tag,
          list_get() );

       if ( exchange_tag_boolean() ) return list;
    }

    boolean exchange_tag_boolean(
       tag,
       list_get() );

    if ( exchange_tag_boolean() ) return list;

    list_next( list );
}
return list;

/* Usage */
boolean exchange_open_tag_boolean(
    const char *EXCHANGE_OFX_TAG,
    LIST *list );

/* Process */
(void)list_rewind( list );

list =
    exchange_tag_seek_list(
       EXCHANGE_OFX_TAG,
       (char *)0 /* stop_tag */,
       list );

return !list_past_end( list );
```

```
/* Usage */
boolean exchange_tag_boolean(
    const char *tag,
    char *list_get );

/* Process */
int string_instr(
       tag /* substr */,
       list_get /* string */,
       1 /* occurrence */ );

if ( string_instr() > -1 )
       return 1;
else
       return 0;

/* Usage */
boolean exchange_open_tag_boolean(
    const char *EXCHANGE_OFX_TAG,
    LIST *list );

/* Process */
(void)list_rewind( list );

list =
    exchange_tag_seek_list(
       EXCHANGE_OFX_TAG,
       (char *)0 /* stop_tag */,
       list );

return !list_past_end( list );

/* Usage */
boolean exchange_tag_boolean(
    const char *tag,
    char *list_get );

/* Process */
int string_instr(
       tag /* substr */,
       list_get /* string */,
       1 /* occurrence */ );

if ( string_instr() > -1 )
       return 1;
else
       return 0;

/* Usage */
char *exchange_financial_institution(
    const char *EXCHANGE_ORG_TAG,
    LIST *list );

/* Process */
(void)list_rewind( list );

list =
    LIST *exchange_tag_seek_list(
       EXCHANGE_ORG_TAG,
       (char *)0 /* stop_tag */,
       list );

if ( list_past_end( list ) ) return NULL;

char *list_get( list );

char *exchange_journal_datum(
    list_get() );

/* Usage */
double exchange_balance_amount(
    const char *EXCHANGE_BALAMT_TAG,
    LIST *list );

/* Process */
(void)list_rewind( list );

list =
    LIST *exchange_tag_seek_list(
       EXCHANGE_BALAMT_TAG,
       (char *)0 /* stop_tag */,
       list );

if ( list_past_end( list ) ) exit( 1 );

char *list_get( list );

char *exchange_journal_datum(
    list_get() );

double string_atof(
    exchange_journal_datum() );
```

EXCHANGE_JOURNAL (1)

```
/* Usage */
EXCHANGE_JOURNAL *exchange_journal_extract(
   LIST *list );

/* Process */
while ( 1 )
{
   list =
      exchange_tag_seek_list(
         EXCHANGE_TRNTYPE_TAG,
         (char *)0 /* stop_tag */,
         list );

   if ( list_past_end( list ) ) return NULL;

   list =
      exchange_tag_seek_list(
         EXCHANGE_DTPOSTED_TAG,
         (char *)0 /* stop_tag */,
         list );

   char *list_get( list );

   char *date_posted =
      char *exchange_journal_datum(
         list_get() );

   list =
      exchange_tag_seek_list(
         EXCHANGE_TRNAMT_TAG,
         (char *)0 /* stop_tag */,
         list );

   char *list_get( list );

   char *amount_string =
      char *exchange_journal_datum(
         list_get() );

   list =
      exchange_tag_seek_list(
         EXCHANGE_NAME_TAG,
         (char *)0 /* stop_tag */,
         list );

   char *list_get( list );

   char *name =
      char *exchange_journal_datum(
         list_get() );

   list =
      exchange_tag_seek_list(
         EXCHANGE_MEMO_TAG,
         EXCHANGE_STMTTRN_END_TAG
            /* stop_tag */,
         list );

   char *list_get( list );

   boolean exchange_tag_boolean(
      EXCHANGE_STMTTRN_END_TAG,
      list_get() );

   if ( !exchange_tag_boolean() )
   {
      char *memo =
         char *exchange_journal_datum(
            list_get() );
   }

   char *exchange_journal_description(
         name,
         memo );

   return
      EXCHANGE_JOURNAL *exchange_journal_new(
         date_posted,
         amount_string,
         exchange_journal_description() );
}

/* Usage */
EXCHANGE_JOURNAL *exchange_journal_new(
   char *date_posted,
   char *amount_string,
   char *description );

/* Process */
EXCHANGE_JOURNAL *exchange_journal_calloc(
   void );

double amount =
   string_atof(
      amount_string );

JOURNAL *journal_calloc();

char *exchange_journal_transaction_date_time(
   char *date_posted );

double exchange_journal_debit_amount(
   char *amount_string );

double exchange_journal_credit_amount(
   char *amount_string );

/* Attributes */
double amount;
char *description;
JOURNAL *journal;
```

JOURNAL

```
char *transaction_date_time
double previous_balance
double debit_amount
double credit_amount
double balance
```

EXCHANGE_JOURNAL (2)

```
/* Usage */
double exchange_journal_begin_amount(
    LIST *exchange_journal_list /* in/out */,
    double exchange_balance_amount() );

/* Process */
LIST *exchange_journal_extract_list(
    LIST *exchange_journal_list );

void journal_propagate_previous_balance_set(
    exchange_journal_extract_list()
       /* journal_list in/out */,
    exchange_balance_amount
       /* end_balance */ );

void journal_propagate_balance_set(
    exchange_journal_extract_list()
       /* journal_list in/out */,
    1 /* accumulate_debit */ );

JOURNAL *list_first(
    exchange_journal_extract_list() );

double list_first()->previous_balance;

/* Usage */
char *exchange_journal_datum(
    char *list_get );

/* Process */
(void)piece( datum[], '>', list_get, 1 );

if ( !*datum ) exit( 1 );

char *strdup( datum );

/* Usage */
int exchange_journal_compare_function(
    EXCHANGE_JOURNAL *exchange_journal_from_list,
    EXCHANGE_JOURNAL *exchange_journal_compare );

/* Process */

/* Usage */
char *exchange_journal_description(
       char *name,
       char *memo );

/* Process */
if ( !memo ) return name;

int strcmp( name, memo );

if ( strcmp() == 0 ) return name;

snprintf(
    char description[],
    sizeof ( description ),
    "% %s",
    name,
    memo );
```

ACCURAL (1)

```
/* Usage */
LIST *accrual_list_fetch( void );

/* Process */
boolean accrual_property_attribute_exists(
   ACCRUAL_TABLE,
   ACCRUAL_PROPERTY_ATTRIBUTE );

char *accrual_select(
   ACCRUAL_SELECT_ATTRIBUTES,
   ACCRUAL_PROPERTY_ATTRIBUTE,
   accrual_property_attribute_exists() );

char *accrual_system_string(
   accrual_select(),
   ACCRUAL_TABLE,
   "1 = 1" /* where */ );

LIST *accrual_system_list(
   accrual_system_string(),
   accrual_property_attribute_exists );

void accrual_list_transaction_set(
   accrual_system_list() /* in/out */ );

/* Usage */
LIST *accrual_system_list(
   char *accrual_system_string(),
   boolean accrual_property_attribute_exists() );

/* Process */
FILE *accrual_input_pipe(
   char *accrual_system_string() );

for input[] in string_input( accrual_input_pipe() )
{
   list_set(
      accrual_list,
      ACCRUAL *
         accrual_parse(
            accrual_property_attribute_exists,
            input ) );
}

void pclose( accrual_input_pipe() );

/* Usage */
ACCRUAL *accrual_fetch(
   char *full_name,
   char *street_address,
   char *accrual_description );

/* Process */
boolean accrual_property_attribute_exists(
   ACCRUAL_TABLE,
   ACCRUAL_PROPERTY_ATTRIBUTE );

char *accrual_select(
   ACCRUAL_SELECT_ATTRIBUTES,
   ACCRUAL_PROPERTY_ATTRIBUTE,
   accrual_property_attribute_exists );

char *accrual_primary_where(
   full_name,
   street_address,
   accrual_description );

char *accrual_system_string(
   accrual_select(),
   ACCRUAL_TABLE,
   accrual_primary_where() );

ACCRUAL *accrual_parse(
   accrual_property_attribute_exists,
   string_pipe_fetch(
      accrual_system_string() ) );

TRANSACTION *accrual_parse()->transaction =
   TRANSACTION *accrual_transaction(
      accrual_parse()->full_name,
      accrual_parse()->street_address,
      accrual_parse()->accrual_description,
      accrual_parse()->
         transaction_increment_date_time,
      accrual_parse()->debit_account,
      accrual_parse()->credit_account,
      accrual_parse()->accrued_daily_amount,
      accrual_parse()->accrued_monthly_amount,
      accrual_parse()->
         rental_property_street_address );

/* Usage */
ACCRUAL *accrual_parse(
   boolean accrual_property_attribute_exists,
   char *input );

/* Process */
ACCRUAL *accrual_new(
   strdup( full_name ),
   strdup( street_address ),
   strdup( accrual_description ) );

/* Increments second each invocation. */
char *transaction_increment_date_time(
   (char *)0 /* transaction_date */ );

/* Usage */
ACCRUAL *accrual_new(
   char *full_name,
   char *street_address,
   char *accrual_description );

/* Process */
ACCRUAL *accrual_calloc( void );

/* Attributes */
char *full_name;
char *street_address;
char *accrual_description;
char *debit_account;
char *credit_account;
double accrued_daily_amount;
double accrued_monthly_amount;
boolean property_attribute_exists;
char *rental_property_street_address;
char *transaction_increment_date_time;
TRANSACTION *transaction;

/* Usage */
void accrual_list_transaction_set(
   LIST *accrual_list() /* in/out */ );

/* Process */
for ACCRUAL *accrual in accrual_list()
{
   TRANSACTION *accrual->transaction =
      TRANSACTION *accrual_transaction(
         accrual->full_name,
         accrual->street_address,
         accrual->accrual_description,
         accrual->
            transaction_increment_date_time,
         accrual->debit_account,
         accrual->credit_account,
         accrual->accrued_daily_amount,
         accrual->accrued_monthly_amount,
         accrual->
            rental_property_street_address );
}
```

ACCRUAL (2)

```
/* Usage */
TRANSACTION *accrual_transaction(
    char *full_name,
    char *street_address,
    char *accrual_description,
    char *transaction_increment_date_time,
    char *debit_account,
    char *credit_account,
    double accrued_daily_amount,
    double accrued_monthly_amount,
    char *rental_property_street_address );

/* Process */
if ( accrued_daily_amount )
{
    TRANSACTION *accrual_daily_transaction(
        full_name,
        street_address,
        accrual_description,
        transaction_increment_date_time,
        debit_account,
        credit_account,
        accrued_daily_amount );
}
else
if ( accrued_monthly_amount )
{
    TRANSACTION *accrual_monthly_transaction(
        full_name,
        street_address,
        accrual_description,
        transaction_increment_date_time,
        debit_account,
        credit_account,
        accrued_monthly_amount,
        rental_property_street_address );
}

/* Usage */
TRANSACTION *accrual_daily_transaction(
    char *full_name,
    char *street_address,
    char *transaction_description,
    char *transaction_increment_date_time,
    char *debit_account,
    char *credit_account,
    double accrued_daily_amount );

/* Process */
int accrual_last_transaction_days_between(
    TRANSACTION_TABLE,
    full_name,
    street_address,
    transaction_increment_date_time,
    debit_account,
    credit_account );

if ( !accrual_last_transaction_days_between() )
{
    return (TRANSACTION *)0;
}

double accrual_daily_accrued_amount(
    double accrued_daily_amount,
    int accrual_last_transaction_days_between() );

char *accrual_memo(
    accrual_description,
    credit_account );

TRANSACTION *transaction_binary(
    full_name,
    street_address,
    transaction_increment_date_time,
    accrual_daily_accrued_amount(),
    accrual_memo(),
    debit_account,
    credit_account );

/* Usage */
TRANSACTION *accrual_monthly_transaction(
    char *full_name,
    char *street_address,
    char *accrual_description,
    char *transaction_increment_date_time(),
    char *debit_account,
    char *credit_account,
    double accrued_monthly_amount,
    char *rental_property_street_address );

/* Process */
char *accrual_max_transaction_date_time(
    TRANSACTION_TABLE,
    full_name,
    street_address,
    debit_account,
    credit_account );

double accrual_monthly_accrue(
    accrual_max_transaction_date_time()
        /* begin_date_string */,
    transaction_increment_date_time()
        /* end_date_string */,
    accrued_monthly_amount );

if ( !accrual_monthly_accrue() ) return NULL;

char *accrual_memo(
    accrual_description,
    credit_account );

TRANSACTION *transaction_binary(
    full_name,
    street_address,
    transaction_increment_date_time,
    accrual_monthly_accrue(),
    accrual_memo(),
    debit_account,
    credit_account );
```

TRANSACTION

ACCRUAL (3)

```
/* Usage */
int accrual_last_transaction_days_between(
    char *TRANSACTION_TABLE,
    char *full_name,
    char *street_address,
    char *transaction_increment_date_time,
    char *debit_account,
    char *credit_account );

char *accrual_max_transaction_date_time(
    TRANSACTION_TABLE,
    full_name,
    street_address,
    debit_account,
    credit_account );

int date_days_between(
    accrual_max_transaction_date_time(),
    transaction_increment_date_time );

/* Usage */
char *accrual_max_transaction_date_time(
    char *TRANSACTION_TABLE,
    char *full_name,
    char *street_address,
    char *debit_account,
    char *credit_account );

/* Process */
char *accrual_journal_transaction_subquery(
    char *JOURNAL_TABLE,
    char *TRANSACTION_TABLE,
    char *debit_account,
    char *credit_account );

/* Usage */
char *accrual_primary_where(
    char *full_name,
    char *street_address,
    char *accrual_description );

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

char *accrual_escape_transaction_description(
    char *transaction_description );

/* Usage */
double accrual_monthly_accrue(
    char *begin_date_string,
    char *end_date_string,
    double accrued_monthly_amount );

/* Process */
DATE *end_date =
    date_yyyy_mm_dd_new(
        end_date_string );

if ( !begin_date_string || !*begin_date_string )
{
    char *accrual_opening_begin_date_string(
        end_date );

    DATE *begin_date =
        date_yyyy_mm_dd_new(
            accrual_opening_begin_date_string() );
}
else
{
    DATE *begin_date =
        date_yyyy_mm_dd_new(
            begin_date_string );
}

date_increment_days(
    begin_date,
    1.0 );

int date_months_between(
    begin_date,
    end_date );

if ( date_months_between() == 0 )
{
    double monthly_accrue =
        accrual_monthly_within_month_accrue(
            DATE *begin_date,
            DATE *end_date,
            double accrued_monthly_amount );
}
else
if ( date_months_between() == 1 )
{
    double monthly_accrue =
        accrual_monthly_next_month_accrue(
            DATE *begin_date,
            DATE *end_date,
            double accrued_monthly_amount );
}
else
{
    double monthly_accrue =
        accrual_monthly_multi_month_accrue(
            DATE *begin_date,
            DATE *end_date,
            double accrued_monthly_amount,
            int date_months_between() );
}
```

ACCRUAL (4)

```
/* Usage */
char *accrual_opening_begin_date_string(
    DATE *end_date );

/* Process */
int date_day_number( end_date );

if ( date_day_number() > 15 )
{
    int date_year_number( end_date );
    int date_month_number( end_date );

    sprintf(
        begin_date_string[],
        "%d-%d-01",
        date_year_number(),
        date_month_number() );
}
else
{
    DATE *begin_date = date_calloc();

    date_copy(
        begin_date,
        end_date );

    date_increment_months(
        begin_date,
        -1 /* months */ );

    date_set_day(
        begin_date,
        1,
        0 /* utc_offset */ );

    strcpy(
        begin_date_string[],
        date_yyyy_mm_dd_display(
            begin_date ) );
}

/* Public */
boolean accrual_property_attribute_exists(
    char *ACCRUAL_TABLE,
    char *ACCRUAL_PROPERTY_ATTRIBUTE );

char *accrual_select(
    char *ACCRUAL_SELECT_ATTRIBUTES,
    char *ACCRUAL_PROPERTY_ATTRIBUTE,
    boolean accrual_property_attribute_exists() );

char *accrual_system_string(
    char *accrual_select(),
    char *ACCRUAL_TABLE,
    char *where );

char *accrual_memo(
    char *accrual_description,
    char *credit_account );

LIST *accrual_transaction_list_extract(
    LIST *accrual_list );

/* Driver */
if ( !execute && with_html )
{
    if ( list_length( accrual_list ) )
    {
        void transaction_list_html_display(
            LIST *accrual_transaction_list_extract(
                accrual_list ) );
    }
    else
    if ( accrual )
    {
        void transaction_html_display(
            accrual->transaction );
    }
}
else
if ( execute )
{
    if ( list_length( accrual_list ) )
    {
        LIST *accrual_transaction_list_extract(
            accrual_list );

        /* May reset transaction->transaction_date_time*/
        void transaction_list_insert(
            accrual_transaction_list_extract()
            1 /* insert_journal_list_boolean */,
            1 /* transaction_lock_boolean */ );
    }
    else
    if ( accrual )
    {
        (void)transaction_stamp_insert(
            accrual->transaction,
            1 /* insert_journal_list_boolean */,
            1 /* transaction_lock_boolean */ );
    }
}
```

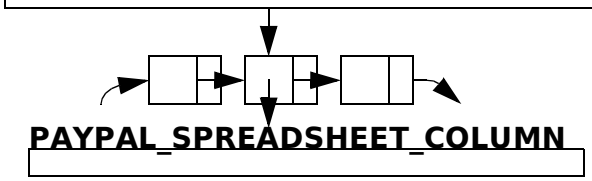
PAYPAL

```
/* Usage */
PAYPAL *paypal_fetch(
    spreadsheet_filename,
    date_label );

/* Process */
PAYPAL_SPREADSHEET *paypal_spreadsheet =
    paypal_spreadsheet_fetch(
       spreadsheet_filename,
       date_label );

/* Attribute */
PAYPAL_SPREADSHEET *paypal_spreadsheet;
```

PAYPAL_SPREADSHEET

PAYPAL_SPREADSHEET

```
/* Usage */
PAYPAL_SPREADSHEET *
    paypal_spreadsheet_fetch(
       char *spreadsheet_filename;
       char *date_label );

/* Process */
PAYPAL_SPREADSHEET *
    paypal_spreadsheet_calloc(
       void );

PAYPAL_SPREADSHEET_SUMMARY *
    paypal_spreadsheet_summary =
       paypal_spreadsheet_summary_new(
          spreadsheet_filename,
          date_label );

if ( paypal_spreadsheet_summary->row_count == 0 )
{
    return this;
}

LIST *paypal_spreadsheet_column_list(
    spreadsheet_filename,
    date_label );

/* Usage */
char *spreadsheet_heading_data(
    LIST *spreadsheet_column_list(),
    char *input_row,
    char *heading_string );

/* Process */

/* Attributes */
char *spreadsheet_filename;
char *date_label;
PAYPAL_SPREADSHEET_SUMMARY *
    paypal_spreadsheet_summary;
LIST *paypal_spreadsheet_column_list;
PAYPAL_SPREADSHEET_UPLOAD_EVENT *
    paypal_spreadsheet_upload_event;
```



PAYPAL_SPREADSHEET_COLUMN

PAYPAL_SPREADSHEET_SUMMARY

PAYPAL_SPREADSHEET_UPLOAD_EVENT

PAYPAL_SPREADSHEET_SUMMARY

```
/* Usage */
PAYPAL_SPREADSHEET_SUMMARY *
    paypal_spreadsheet_summary_new(
       char *spreadsheet_filename,
       char *date_label );

/* Process */
PAYPAL_SPREADSHEET_SUMMARY *
    paypal_spreadsheet_summary_calloc(
       void );

int paypal_spreadsheet_column_date_piece(
    spreadsheet_filename,
    date_label );

if ( paypal_spreadsheet_column_date_piece() == -1 )
    return this;

FILE *appaserver_input_file(
    spreadsheet_filename );

for input[] in string_input( appaserver_input_file() )
{
    piece_quote(
       date_american_string[],
       input,
       paypal_spreadsheet_column_date_piece() );

char *date_convert_source_american(
   date_convert_international
      /* destination_enum */,
   date_american_string );

if ( !date_convert_source_american() ) continue;

row_count++;

if ( !*minimum_date_international )
{
   strcpy(
      minimum_date_international,
      date_convert_source_american() );

   strcpy(
      maximum_date_international,
      date_convert_source_american() );
}
else
{
   if ( strcmp(
         date_convert_source_american(),
         minimum_date_international ) < 0 )
   {
      strcpy(
         minimum_date_international,
         date_convert_source_american() );
   }
   else
   if ( strcmp(
         date_convert_source_american(),
         maximum_date_international ) > 0 )
   {
      strcpy(
         maximum_date_international,
         date_convert_source_american() );
   }
}

if (   date_convert_source_american() !=
       date_american_string )
{
   free( date_convert_source_american() );
}
}
fclose( appaserver_input_file() );

/* Attributes */
int paypal_spreadsheet_column_date_piece;
int row_count;
char minimum_date_international[ 11 ];
char maximum_date_international[ 11 ];
```

PAYPAL_SPREADSHEET_COLUMN (1)

```
/* Usage */
LIST *paypal_spreadsheet_column_list(
    char *spreadsheet_filename,
    char *date_label );

/* Process */
char *paypal_spreadsheet_column_header_row_string(
    spreadsheet_filename,
    date_label );

if ( !paypal_spreadsheet_column_header_row_string() )
    return NULL;

LIST *column_list = list_new();

for ( int p = 0;
      piece_quote_comma(
        heading_string[],
        paypal_spreadsheet_header_row_string(),
        p );
      p++ )
{
    PAYPAL_SPREADSHEET_COLUMN *
      paypal_spreadsheet_column_new(
        strdup( heading_string ),
        p /* paypal_spreadsheet_column_piece */ );

    list_set(
      column_list,
      paypal_spreadsheet_column_new() );
}

/* Usage */
PAYPAL_SPREADSHEET_COLUMN *
    paypal_spreadsheet_column_new(
      char *heading_string,
      int paypal_spreadsheet_column_piece() );

/* Process */
PAYPAL_SPREADSHEET_COLUMN *
    paypal_spreadsheet_column_calloc(
      void );

/* Usage */
PAYPAL_SPREADSHEET_COLUMN *
    paypal_spreadsheet_column_heading_seek(
      LIST *paypal_spreadsheet_column_list(),
      char *heading_string );

/* Process */
for PAYPAL_SPREADSHEET_COLUMN *
    paypal_spreadsheet_column in
      paypal_spreadsheet_column_list
{
    char *string_remove_control(
      paypal_spreadsheet_column->heading_string );

    if ( string_strcmp(
          string_remove_control(),
          heading_string ) == 0 )
    {
      return paypal_spreadsheet_column;
    }
}
return NULL;

/* Usage */
int paypal_spreadsheet_column_piece(
    char *heading_line,
    char *heading_string );

/* Process */
for (  int p = 0;
      piece_quote_comma(
        piece_buffer[],
        heading_line,
        p );
      p++ )
{
    if ( string_strcmp(
          piece_quote_comma(),
          heading_string ) == 0 )
    {
      return p;
    }
}
return -1;

/* Usage */
char *paypal_spreadsheet_column_header_row_string(
    char *spreadsheet_filename,
    char *date_label );

/* Process */
FILE *appaserver_input_file(
    spreadsheet_filename );

for char header_buffer[] in
    string_input( appaserver_input_file() )
{
    boolean
    paypal_spreadsheet_column_header_boolean(
      date_label,
      header_buffer );

    if ( paypal_spreadsheet_column_header_boolean() )
    {
      fclose( appaserver_input_file() );
      return strdup( header_buffer );
    }
}
fclose( appaserver_input_file() );
return NULL;

/* Usage */
boolean paypal_spreadsheet_column_header_boolean(
    char *date_label,
    char *header_buffer );

/* Process */
int instr( date_label, header_buffer );

return ( instr() > -1 );

/* Attributes */
char *heading_string;
int paypal_spreadsheet_column_piece;
```

PAYPAL_SPREADSHEET_COLUMN (2)

```
/* Usage */
char *paypal_spreadsheet_column_heading_data(
    LIST *paypal_spreadsheet_column_list(),
    char *input,
    char *heading_string );

/* Process */
PAYPAL_SPREADSHEET_COLUMN *
    paypal_spreadsheet_column_heading_seek(
       paypal_spreadsheet_column_list,
       heading_string );

if ( !paypal_spreadsheet_column_heading_seek() )
    return NULL;

char *piece_quote(
    data[] /* destination */,
    input /* source */,
    paypal_spreadsheet_column_heading_seek()->
       paypal_spreadsheet_column_piece );

if ( !piece_quote() ) return NULL;

return
strdup( data );

/* Usage */
PAYPAL_SPREADSHEET_COLUMN *
    paypal_spreadsheet_column_seek(
       LIST *paypal_spreadsheet_column_list(),
       char *heading_string );

/* Process */
for PAYPAL_SPREADSHEET_COLUMN *
    paypal_spreadsheet_column in
       paypal_spreadsheet_column_list
{
    char *string_remove_control(
       paypal_spreadsheet_column->
          heading_string );

    int string_strcmp(
       string_remove_control(),
       heading_string );

    if ( string_strcmp() == 0 )
    {
       return paypal_spreadsheet_column;
    }
}
return NULL;

/* Usage */
char *paypal_spreadsheet_column_cell_extract(
    char *input,
    int paypal_spreadsheet_column_piece() );

/* Process */
int piece_quote_comma(
    cell_extract[],
    input,
    paypal_spreadsheet_column_piece );

if ( piece_quote_comma() )
    return strdup( cell_extract );
else
    return NULL;

/* Usage */
int paypal_spreadsheet_column_date_piece(
    char *spreadsheet_filename,
    char *date_label );

/* Process */
char *paypal_spreadsheet_column_header_row_string(
    spreadsheet_filename,
    date_label );

if ( !paypal_spreadsheet_column_header_row_string() )
    return -1;

int piece_quote_comma_seek(
    paypal_spreadsheet_column_header_row_string(),
    date_label );
```

PAYPAL_SPREADSHEET_UPLOAD_EVENT

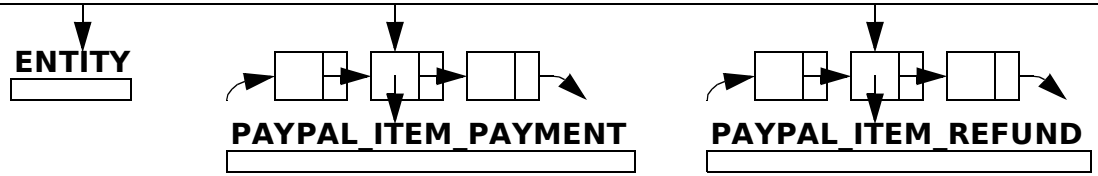
```
/* Usage */
PAYPAL_SPREADSHEET_UPLOAD_EVENT *
    paypal_spreadsheet_upload_event_new(
       char *spreadsheet_name,
       char *spreadsheet_filename );

/* Process */
PAYPAL_SPREADSHEET_UPLOAD_EVENT *
    paypal_spreadsheet_upload_event_calloc(
       void );

char *paypal_spreadsheet_upload_event_date_time();

char *paypal_spreadsheet_upload_event_sha256sum(
    spreadsheet_filename );

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

/* Process */
char *date_time19_now(
    date_utc_offset() );

/* Usage */
char *paypal_spreadsheet_upload_event_sha256sum(
    char *spreadsheet_filename );

/* Process */
boolean timlib_file_exists( spreadsheet_filename );

if ( !timlib_file_exists() ) return NULL;

sprintf(
    system_string[],
    "cat \"%s\" |"
    "sha256sum |"
    "column.e 0",
    spreadsheet_filename );

char *string_pipe_fetch( system_string );

/* Driver */
if ( !paypal_spreadsheet_upload_event->sha256sum )
{
    exit( 1 );
}

/* Attributes */
char *spreadsheet_name;
char *spreadsheet_filename;
char *date_time;
char *sha256sum;
```

PAYPAL_DATASET

```
/* Usage */
PAYPAL_DATASET *paypal_dataset_parse(
    LIST *paypal_spreadsheet_column_list,
    char *input );

/* Process */
PAYPAL_DATASET *paypal_dataset_calloc(
    void );

char *paypal_dataset_date_A(
    paypal_spreadsheet_column_list,
    input,
    “Date” /* heading_label */ );

if ( !paypal_dataset_date_A() ) return NULL;

char *paypal_dataset->time_B =
    paypal_spreadsheet_heading_data(
       paypal_spreadsheet_column_list,
       input,
       “Time” /* heading_label */ );

if ( !paypal_dataset->time_B ) return NULL;

...

/* Usage */
char *paypal_dataset_date_A(
    LIST *paypal_spreadsheet_column_list,
    char *input,
    char *heading_label );

/* Process */
char *paypal_spreadsheet_heading_data(
    paypal_spreadsheet_column_list,
    input,
    heading_label );

/* Attributes */
char *date_A;
char *time_B;
char *timezone_C;
char *full_name_D;
char *transaction_type_E;
char *transaction_status_F;
char *currency_G;
char *gross_revenue_H;
char *transaction_fee_I;
char *net_revenue_J;
char *from_email_address_K;
char *to_email_address_L;
char *transaction_ID_M;
char *shipping_address_N;
char *address_status_O;
char *item_title_P;
char *item_ID_Q;
char *shipping_handling_amount_R;
char *insurance_amount_S;
char *sales_tax_T;
char *option_1_name_U;
char *option_1_value_V;
char *option_2_name_W;
char *option_2_value_X;
char *reference_txn_ID_Y;
char *invoice_number_Z;
char *custom_number_AA;
char *quantity_AB;
char *receipt_ID_AC;
char *balance_AD;
char *address_line_1_AE;
char *address_2_district_neighborhood_AF;
char *town_city_AG;
char *state_province_AH;
char *zip_code_AI;
char *country_AJ;
char *contact_phone_number_AK;
char *subject_AL;
char *note_AM;
char *country_code_AN;
char *balance_impact_AO;
```

PAYPAL_ITEM_LIST

```
/* Usage */
PAYPAL_ITEM_LIST *paypal_item_new(
    char *item_data );

/* Process */
PAYPAL_ITEM_LIST *paypal_item_list_calloc(
    void );

/* Attributes */
char *item_data;
char *entity_delimited_item_title_P;
char *transaction_type_E;
ENTITY *benefit_entity;
LIST *paypal_item_payment_list;
LIST *paypal_item_refund_list;
```



PAYPAL_ITEM_PAYMENT

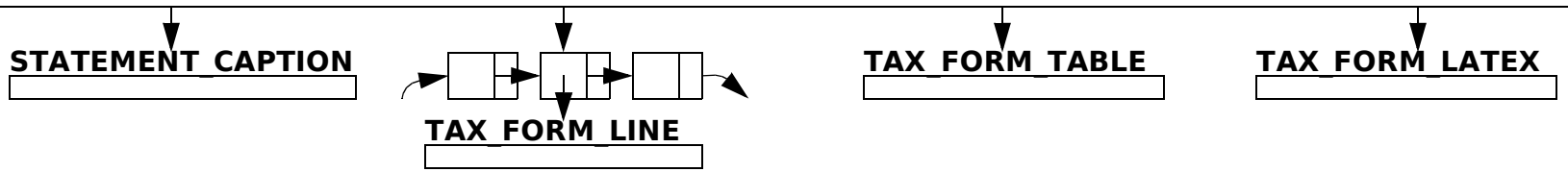
```
/* Usage */
PAYPAL_ITEM_PAYMENT *paypal_item_payment_new(
    char *item_data );

/* Process */
PAYPAL_ITEM_PAYMENT *paypal_item_payment_calloc(
    void );

/* Attributes */
char *entity_delimited_item_title_P;
char *transaction_type_E;
double expected revenue;
char *item_data;
double item_value;
double item_fee;
double item_gain;
boolean taken;
```

PAYPAL_ITEM_REFUND

```
/* Usage */
PAYPAL_ITEM_REFUND *paypal_item_refund_new(
    char *item_data );

/* Process */
PAYPAL_ITEM_REFUND *paypal_item_refund_calloc(
    void );

/* Attributes */
char *entity_delimited_item_title_P;
char *transaction_type_E;
double expected revenue;
char *item_data;
double item_value;
double item_fee;
double item_gain;
boolean taken;
```

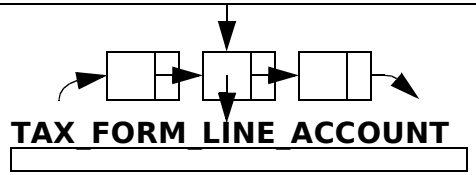
TAX_FORM

```
/* Usage */
TAX_FORM *tax_form_fetch(
    char *application_name,
    char *process_name,
    char *appaserver_parameter_data_directory,
    char *tax_form_name,
    int tax_year,
    int fiscal_begin_month /* zero or one based */,
    char *output_medium_string );

/* Process */
TAX_FORM *tax_form_calloc(
    void );

enum statement_output_medium
    statement_output_medium =
       statement_resolve_output_medium(
          char *output_medium_string );

char *tax_form_fiscal_begin_date(
    int tax_year,
    int fiscal_begin_month );

char *tax_form_fiscal_end_date(
    int tax_year,
    int fiscal_begin_month );

STATEMENT_CAPTION *statement_caption =
    statement_caption_new(
       application_name,
       process_name,
       tax_form_fiscal_begin_date,
       tax_form_fiscal_end_date_string );

LIST *tax_form_line_list(
    tax_form_name,
    tax_form_fiscal_begin_date(),
    tax_form_fiscal_end_date() );

if ( statement_output_medium ==
    statement_output_table )
{
    TAX_FORM_TABLE *tax_form_table =
       tax_form_table_new(
          tax_form_name,
          statement_caption,
          tax_form_line_list() );

    if ( !tax_form_table ) return NULL;
}
else
if ( statement_output_medium ==
    statement_output_PDF )
{
    TAX_FORM_LATEX *tax_form_latex =
       tax_form_latex_new(
          application_name,
          process_name,
          appaserver_parameter_data_directory,
          tax_form_name,
          statement_caption->title,
          statement_caption->sub_title,
          statement_caption->date_now16,
          tax_form_line_list() );

    if ( !tax_form_latex ) return NULL;
}

/* Usage */
char *tax_form_primary_where(
    char *tax_form_name );

/* Process */

/* Driver */
if ( tax_form->tax_form_table )
{
    void html_table_list_output(
          tax_form->
             tax_form_table->
             html_table_list,
          HTML_TABLE_ROWS_BETWEEN_HEADING );
}
else
if ( tax_form->tax_form_latex )
{
    void latex_table_list_output(
          tax_form->
             tax_form_latex->
             statement_link->
             tex_filename,
          tax_form->
             tax_form_latex->
             statement_link->
             tex_filename,
          tax_form->
             tax_form_latex->
             statement_link->
             pdf_anchor_html,
          tax_form->
             tax_form_latex->
             appaserver_link_working_directory,
          tax_form->
             tax_form_latex->
             latex,
          tax_form->
             tax_form_latex->
             latex_table_list );
}

/* Attributes */
enum statement_output_medium
    statement_output_medium;
char *fiscal_begin_date;
char *fiscal_end_date;
LIST *tax_form_line_list;
TAX_FORM_TABLE *tax_form_table;
```



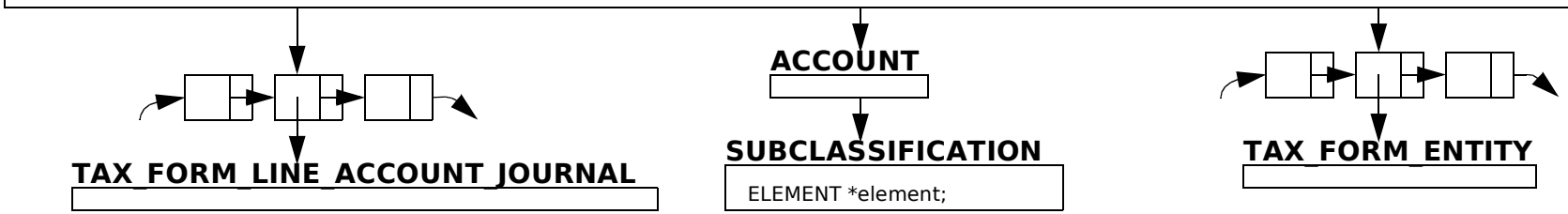
TAX_FORM_LINE

```
/* Usage */
LIST *tax_form_line_list(
    char *tax_form_name,
    char *tax_form_fiscal_begin_date(),
    char *tax_form_fiscal_end_date() );

/* Process */
char *tax_form_primary_where(
    tax_form_name );

char *tax_form_line_system_string(
    char *TAX_FORM_LINE_SELECT,
    char *TAX_FORM_LINE_TABLE,
    char *tax_form_primary_where() );

LIST *tax_form_line_system_list(
    tax_form_name,
    tax_form_fiscal_begin_date,
    tax_form_fiscal_end_date,
    tax_form_line_system_string() );

/* Usage */
LIST *tax_form_line_system_list(
    tax_form_name,
    char *tax_form_fiscal_begin_date(),
    char *tax_form_fiscal_end_date(),
    char *tax_form_line_system_string() );

/* Process */
FILE *tax_form_line_input_pipe(
    char *tax_form_line_system_string );

for input[] in string_input( tax_form_line_input_pipe() )
{
    list_set(
       list,
       tax_form_line_parse(
          tax_form_name,
          tax_form_fiscal_begin_date,
          tax_form_fiscal_end_date,
          input ) );
}

pclose( tax_form_line_input_pipe() );

/* Usage */
TAX_FORM_LINE *tax_form_line_parse(
    char *tax_form_name,
    char *tax_form_fiscal_begin_date,
    char *tax_form_fiscal_end_date,
    char *input );

/* Process */
TAX_FORM_LINE *tax_form_line_calloc( void );

LIST *tax_form_line_account_list(
    tax_form_name,
    tax_form_fiscal_begin_date,
    tax_form_fiscal_end_date,
    tax_form_line->string );

double tax_form_line_total(
    LIST *tax_form_line_account_list() );

/* Usage */
char *tax_form_line_where(
    char *tax_form_name,
    char *tax_form_line_string );

/* Process */

/* Attributes */
char *tax_form_name;
char *string;
char *description;
LIST *tax_form_line_account_list;
double total;
```



TAX_FORM_LINE_ACCOUNT

```
/* Usage */
LIST *tax_form_line_account_list(
    char *tax_form_name,
    char *tax_form_fiscal_begin_date(),
    char *tax_form_fiscal_end_date(),
    char *tax_form_line_string );

/* Process */
char *tax_form_line_where(
    tax_form_name,
    tax_form_line_string );

char *tax_form_line_account_system_string(
    char *TAX_FORM_LINE_ACCOUNT_SELECT,
    char *TAX_FORM_LINE_ACCOUNT_TABLE,
    char *tax_form_line_where() );

LIST *tax_form_line_account_system_list(
    tax_form_name,
    tax_form_fiscal_begin_date,
    tax_form_fiscal_end_date,
    tax_form_line_string,
    tax_form_line_account_system_string() );

/* Usage
LIST *tax_form_line_account_system_list(
    char *tax_form_name,
    char *tax_form_fiscal_begin_date,
    char *tax_form_fiscal_end_date,
    char *tax_form_line_string,
    char *tax_form_line_account_system_string() );

/* Process */
FILE *tax_form_line_account_input_pipe(
    char *tax_form_line_account_system_string );

for input[] in string_input(
    tax_form_line_account_input_pipe() )
{
    list_set(
        list,
        tax_form_line_account_parse(
            tax_form_name,
            tax_form_fiscal_begin_date,
            tax_form_fiscal_end_date,
            tax_form_line_string,
            input ) );
}

pclose( tax_form_line_account_input_pipe() );

/* Usage */
TAX_FORM_LINE_ACCOUNT *
    tax_form_line_account_parse(
        char *tax_form_name,
        char *tax_form_fiscal_begin_date,
        char *tax_form_fiscal_end_date,
        char *tax_form_line_string,
        char *input );

/* Process */
TAX_FORM_LINE_ACCOUNT *
    tax_form_line_account_calloc(
        void );

ACCOUNT *account =
    account_fetch(
        account_name,
        1 /* fetch_subclassification */,
        1 /* fetch_element */ ),

boolean subclassification_current_liability_boolean(
    SUBCLASSIFICATION_CURRENT_LIABILITY,
    account->
        subclassification->
        subclassification_name );

boolean subclassification_receivable_boolean(
    SUBCLASSIFICATION_RECEIVABLE,
    account->
        subclassification->
        subclassification_name );

LIST *tax_form_line_account_journal_list(
    tax_form_fiscal_begin_date,
    tax_form_fiscal_end_date,
    TRANSACTION_DATE_PRECLOSE_TIME,
    account_name,
    account->
        subclassification->
        element->
        accumulate_debit
            /* element_accumulate_debit */,
    subclassification_current_liability_boolean(),
    subclassification_receivable_boolean() );

double tax_form_line_account_journal_list_total(
    tax_form_line_account_journal_list() );

LIST *tax_form_entity_list(
    tax_form_line_account_journal_list() );

/* Attributes */
char *tax_form_name;
char *tax_form_line_string;
char *account_name;
ACCOUNT *account;
boolean subclassification_current_liability_boolean;
boolean subclassification_receivable_boolean;
LIST *journal_list;
double tax_form_line_account_journal_list_total;
LIST *tax_form_entity_list;
```

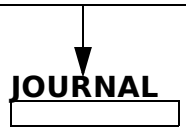


TAX_FORM_LINE_ACCOUNT_JOURNAL

```
/* Usage */
LIST *tax_form_line_account_journal_list(
    char *tax_form_fiscal_begin_date(),
    char *tax_form_fiscal_end_date(),
    const char *TRANSACTION_DATE_PRECLOSE_TIME,
    char *account_name,
    boolean element_accumulate_debit,
    boolean subclassification_current_liability_boolean(),
    boolean subclassification_receivable_boolean() );

/* Process */
LIST *journal_tax_form_list(
     tax_form_fiscal_begin_date,
     tax_form_fiscal_end_date,
     TRANSACTION_DATE_PRECLOSE_TIME,
     account_name );

for JOURNAL *journal in journal_tax_form_list()
{
    list_set(
       list,
       tax_form_line_account_journal_new(
          element_accumulate_debit,
          subclassification_current_liability_boolean,
          subclassification_receivable_boolean,
          journal ) );
}
```

```
/* Usage */
TAX_FORM_LINE_ACCOUNT_JOURNAL *
    tax_form_line_account_journal_new(
       boolean element_accumulate_debit,
       subclassification_current_liability_boolean,
       boolean subclassification_receivable_boolean,
       JOURNAL *journal );

/* Process */
TAX_FORM_LINE_ACCOUNT_JOURNAL *
    tax_form_line_account_journal_calloc(
       void );

double tax_form_line_account_journal_amount(
    element_accumulate_debit,
    subclassification_current_liability_boolean,
    subclassification_receivable_boolean,
    journal->debit_amount,
    journal->credit_amount );

/* Usage */
double tax_form_line_account_journal_amount(
     boolean element_accumulate_debit,
     boolean subclassification_current_liability_boolean,
     boolean subclassification_receivable_boolean,
     double journal->debit_amount,
     double journal->credit_amount );
```

```
/* Process */
if ( subclassification_current_liability_boolean )
{
    amount = debit_amount;
}
else
if ( subclassification_receivable_boolean )
{
    amount = credit_amount;
}
else
{
    amount =
       journal_amount(
          debit_amount,
          credit_amount,
          element_accumulate_debit );
}

/* Usage */
double tax_form_line_account_journal_list_total(
    LIST *tax_form_line_account_journal_list() );

/* Process */

/* Attributes */
JOURNAL *journal;
double amount;
```



TAX_FORM_ENTITY

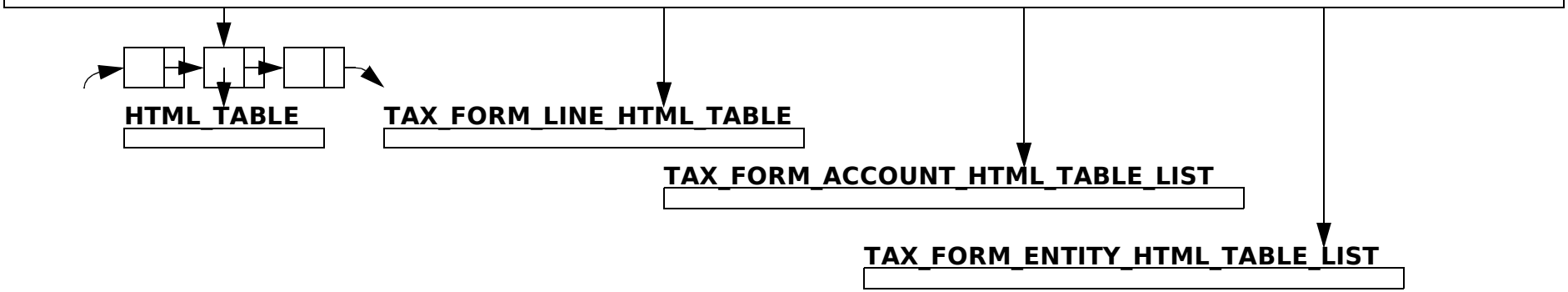
```
/* Usage */
LIST *tax_form_entity_list(
    LIST *tax_form_line_account_journal_list() );

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

for TAX_FORM_LINE_ACCOUNT_JOURNAL *journal in
     tax_form_line_account_journal_list
{
     void tax_form_entity_getset(
        list,
        journal->journal->full_name,
        journal->amount );
}
```

```
/* Usage */
void tax_form_entity_getset(
     LIST *list,
     char *full_name,
     double journal_amount );

/* Process */
for TAX_FORM_ENTITY *tax_form_entity in list
{
     if ( strcmp(
        tax_form_entity->entity->full_name,
        full_name ) == 0 )
     {
        tax_form_entity->total += journal_amount;
        return;
     }
}
```

```
TAX_FORM_ENTITY *tax_form_entity_calloc( void );

ENTITY *entity =
     entity_new(
        full_name,
        (char *)0 /* street_address */ );

total = journal_amount;

list_set( list, tax_form_entity_calloc() );

/* Attributes */
ENTITY *entity;
double total;
```

TAX_FORM_TABLE

```
/* Usage */
TAX_FORM_TABLE *tax_form_table_new(
    char *tax_form_name,
    STATEMENT_CAPTION *statement_caption,
    LIST *tax_form_line_list() );

/* Process */
TAX_FORM_TABLE *tax_form_table_calloc( void );

LIST *html_table_list = list_new();

TAX_FORM_LINE_HTML_TABLE *
   tax_form_line_html_table =
      tax_form_line_html_table_new(
         tax_form_name,
         statement_caption->sub_title,
         tax_form_line_list );

if ( !tax_form_line_html_table ) return NULL;

list_set(
   html_table_list,
   tax_form_line_html_table->html_table );

TAX_FORM_ACCOUNT_HTML_LIST *
    tax_form_account_html_list =
       tax_form_account_html_list_new(
          tax_form_line_list );

list_set_list(
   html_table_list,
   tax_form_account_html_list->
      html_table_list );

TAX_FORM_ENTITY_HTML_LIST *
    tax_form_entity_html_list =
       tax_form_entity_html_list_new(
          tax_form_line_list );

list_set_list(
   html_table_list,
   tax_form_entity_html_list->
      html_table_list );

/* Attributes */
LIST *html_table_list;
TAX_FORM_LINE_HTML_TABLE *
    tax_form_line_html_table;
TAX_FORM_ACCOUNT_HTML_TABLE_LIST *
    tax_form_account_html_table_list;
TAX_FORM_ENTITY_HTML_TABLE_LIST *
    tax_form_entity_html_table_list;
```

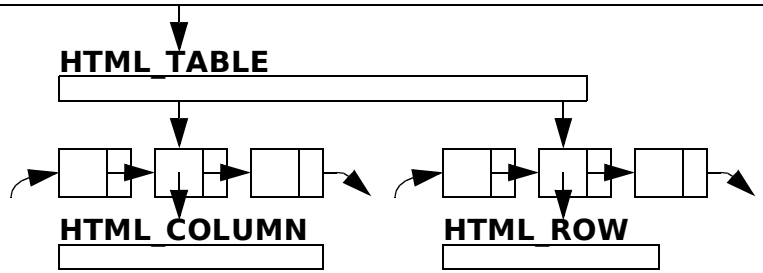


TAX_FORM_LINE_HTML_TABLE

```
/* Usage */
TAX_FORM_LINE_HTML_TABLE *
   tax_form_line_html_table_new(
      char *tax_form_name,
      char *statement_caption_sub_title,
      LIST *tax_form_line_list );

/* Process */
TAX_FORM_LINE_HTML_TABLE *
   tax_form_line_html_table_calloc(
      void );

char *tax_form_line_html_table_caption(
   char *tax_form_name,
   char *statement_caption_sub_title );

HTML_TABLE *html_table =
    html_table_new(
       (char *)0 /* title */,
       (char *)0 /* sub_title */,
       tax_form_line_html_table_caption()
          /* sub_sub_title */ );

html_table->column_list =
    LIST *tax_form_line_html_table_column_list();

html_table->row_list =
    LIST *tax_form_line_html_table_row_list(
       tax_form_line_list );

if ( !list_length( html_table->row_list ) ) return NULL;

/* Usage */
LIST *tax_form_line_html_table_column_list(
   void );
```

```
/* Process */
LIST *column_list = list_new();

list_set(
   column_list,
   HTML_COLUMN *html_column_new(
      "tax_form_line" /* heading */,
      0 /* right_justify_flag */ ) );

list_set(
   column_list,
   HTML_COLUMN *html_column_new(
      "tax_form_description" /* heading */,
      0 /* not right_Justified_flag */ ) );

list_set(
   column_list,
   HTML_COLUMN *html_column_new(
      "balance",
      1 /* right_justified_flag */ ) );

/* Usage */
LIST *tax_form_line_html_table_row_list(
    LIST *tax_form_line_list );

/* Process */
for TAX_FORM_LINE *tax_form_line in tax_form_line_list
{
   if ( !tax_form_line->total ) continue;

   list_set(
      row_list,
      HTML_ROW *tax_form_line_html_table_row(
         tax_form_line ) );
}
```

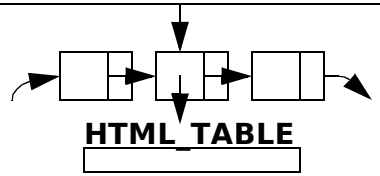
```
/* Usage */
HTML_ROW *tax_form_line_html_table_row(
   TAX_FORM_LINE *tax_form_line );

/* Process */
LIST *cell_list = list_new;

list_set(
   cell_list,
   html_cell_new(
      tax_form_line->tax_form_line_string,
      0 /* not large_boolean */,
      0 /* not bold_boolean */ ) );

list_set(
   cell_list,
   html_cell_new(
      tax_form_line->tax_form_description,
      0 /* not large_boolean */,
      0 /* not bold_boolean */ ) );

char *string_commas_double(
   tax_form_line->total,
   0 /* decimal_count */ );

list_set(
   cell_list,
   html_cell_new(
      strdup( string_commas_double() ),
      0 /* not large_boolean */,
      0 /* not bold_boolean */ ) );

HTML_ROW *html_row_new( cell_list );

/* Attribute */
HTML_TABLE *html_table;
```



TAX_FORM_ACCOUNT_HTML_LIST

```
/* Usage */
TAX_FORM_ACCOUNT_HTML_LIST *
    tax_form_account_html_list_new(
        LIST *tax_form_line_list );

/* Process */
TAX_FORM_ACCOUNT_HTML_LIST *
    tax_form_account_html_list_calloc(
        void );
```

```
LIST *html_table_list = list_new();

for TAX_FORM_LINE *tax_form_line in tax_form_line_list
{
    if ( !tax_form_line->total ) continue;

    list_set(
        html_table_list,
        HTML_TABLE *tax_form_account_html_table(
            tax_form_line->string,
            tax_form_line->
                tax_form_line_account_list,
            tax_form_line->description,
            tax_form_line->total ) );
}

/* Attribute */
LIST *html_table_list;
```

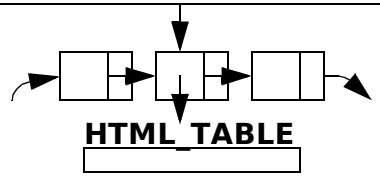


TAX_FORM_ACCOUNT_HTML

```
/* Usage */
HTML_TABLE *tax_form_account_html_table(
    char *tax_form_line_string,
    LIST *tax_form_line_account_list,
    char *tax_form_line_description,
    double tax_form_line_total );

/* Process */
char *tax_form_account_html_caption(
    char *tax_form_line_string,
    char *tax_form_line_description,
    double tax_form_line_total );

HTML_TABLE *html_table =
     html_table_new(
        (char *)0 /* title */,
        (char *)0 /* sub_title */,
        tax_form_account_html_caption()
           /* sub_sub_title */ );

html_table->column_list =
     LIST *tax_form_account_html_column_list();

html_table->row_list =
     LIST *tax_form_account_html_row_list(
        tax_form_line_account_list );

/* Usage */
LIST *tax_form_account_html_column_list(
    void );

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

list_set(
    column_list,
    HTML_COLUMN *html_column_new(
       “account_name“ /* heading */,
       0 /* right_justify_flag */ ) );

list_set(
    column_list,
    HTML_COLUMN *html_column_new(
       “total” /* heading */,
       1 /* right_justified_flag */ ) );

/* Usage */
LIST *tax_form_account_html_row_list(
     LIST *tax_form_line_account_list );

/* Process */
for TAX_FORM_LINE_ACCOUNT *
       tax_form_line_account in
          tax_form_line_account_list
{
    if ( !tax_form_line_account->journal_list_total )
          continue;

    list_set(
       row_list,
          tax_form_line_account_html_row(
             tax_form_line_account ) );
}

/* Usage */
HTML_ROW *tax_form_line_account_html_row(
    TAX_FORM_LINE_ACCOUNT *tax_form_line_account );

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

list_set(
    cell_list,
    html_cell_new(
       tax_form_line_account->account_name,
       0 /* not large_boolean */,
       0 /* not bold_boolean */ ) );

char *string_commas_double(
    tax_form_line_account->journal_list_total,
    0 /* decimal_count */ );

list_set(
    cell_list,
    html_cell_new(
       strdup(
          string_commas_double() ),
       0 /* not large_boolean */,
       0 /* not bold_boolean */ ) );
```

TAX_FORM_ENTITY_HTML_LIST

```
/* Usage */
TAX_FORM_ENTITY_HTML_LIST *
    tax_form_entity_html_list_new(
        LIST *tax_form_line_list );

/* Process */
TAX_FORM_ENTITY_HTML_LIST *
    tax_form_entity_html_list_calloc(
        void );

LIST *html_table_list = list_new();

for TAX_FORM_LINE *tax_form_line in tax_form_line_list
{
    if ( !tax_form_line->total ) continue;

    for TAX_FORM_LINE_ACCOUNT *
        tax_form_line_account in
            tax_form_line->
                tax_form_line_account_list
    {
        if ( !tax_form_line_account->journal_list_total )
            continue;

        list_set(
            html_table_list,
            HTML_TABLE *tax_form_entity_html_table(
                tax_form_line->string,
                tax_form_line_account->
                    account_name,
                tax_form_line_account->
                    journal_list_total,
                tax_form_line_account->
                    tax_form_entity_list ) );
    }
}

/* Attribute */
LIST *html_table_list;
```

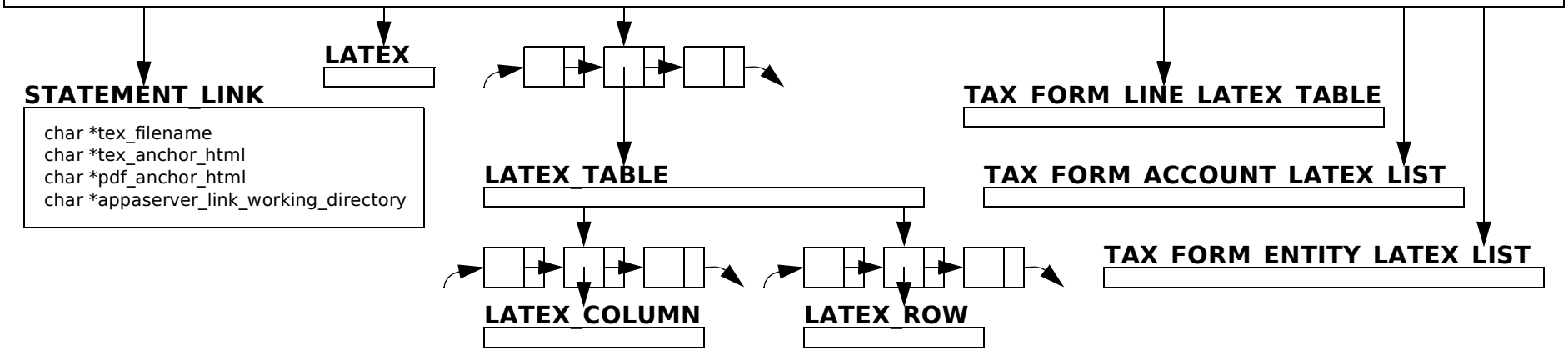


TAX_FORM_ENTITY_HTML

```
/* Usage */
HTML_TABLE *tax_form_entity_html_table(
    char *tax_form_line_string,
    char *account_name,
    double journal_list_total,
    LIST *tax_form_entity_list );

/* Process */
char *tax_form_entity_html_caption(
    char *tax_form_line_string,
    char *account_name,
    double journal_list_total );

HTML_TABLE *html_table =
     html_table_new(
        (char *)0 /* title */,
        (char *)0 /* sub_title */,
        tax_form_entity_html_caption()
           /* sub_sub_title */ );

html_table->column_list =
     LIST *tax_form_entity_html_column_list();

html_table->row_list =
     LIST *tax_form_entity_html_row_list(
        tax_form_entity_list );

/* Usage */
LIST *tax_form_entity_html_column_list(
    void );

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

list_set(
    column_list,
    html_column_new(
       “full_name“,
       0 /* right_justify_flag */ ) );

list_set(
    column_list,
    html_column_new(
       “total”,
       1 /* right_justified_flag */ ) );

/* Usage */
LIST *tax_form_entity_html_row_list(
     LIST *tax_form_entity_list );

/* Process */
for TAX_FORM_ENTITY *
       tax_form_entity in
          tax_form_entity_list
{
    if ( !tax_form_entity->total ) continue;

    list_set(
       row_list,
       tax_form_entity_html_row(
          tax_form_entity ) );
}

/* Usage */
HTML_ROW *tax_form_entity_html_row(
    TAX_FORM_ENTITY *tax_form_entity );

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

list_set(
    cell_list,
    html_cell_new(
       tax_form_entity->entity->full_name,
       0 /* not large_boolean */,
       0 /* not bold_boolean */ ) );

char *string_commas_double(
    tax_form_entity->total,
    0 /* decimal_count */ );

list_set(
    cell_list,
    html_cell_new(
       strdup(
          string_commas_double() ),
       0 /* not large_boolean */,
       0 /* not bold_boolean */ ) );

HTML_ROW *html_row_new( cell_list );
```

TAX_FORM_LATEX

```
/* Usage */
TAX_FORM_LATEX *tax_form_latex_new(
    char *application_name,
    char *process_name,
    char *appaserver_parameter_data_directory,
    char *tax_form_name,
    char *statement_caption_title,
    char *statement_caption_sub_title,
    char *statement_caption_date_now16,
    LIST *tax_form_line_list() );

/* Process */
TAX_FORM_LATEX *tax_form_latex_calloc(
    void );

STATEMENT_LINK *statement_link =
   statement_link_new(
      application_name,
      process_name,
      appaserver_parameter_data_directory,
      statement_caption_date_now16
         /* transaction_date_begin_date_string */,
      (char *)0
          /* end_date_string */,
      getpid() /* process_id */ );

LATEX *latex =
   latex_new(
      statement_link->tex_filename,
      statement_link->
         appaserver_link_working_directory,
      0 /* not landscape_boolean */,
      (char *)0 /* logo_filename */ );

LIST *latex_table_list = list_new();

TAX_FORM_LINE_LATEX_TABLE *
   tax_form_line_latex_table =
      tax_form_line_latex_table_new(
         tax_form_name,
         statement_caption_title,
         statement_caption_sub_title,
         tax_form_line_list );

if ( !tax_form_line_latex_table ) return NULL;

list_set(
   latex_table_list,
   tax_form_line_latex_table->latex_table );

TAX_FORM_ACCOUNT_LATEX_LIST *
    tax_form_account_latex_list =
       tax_form_account_latex_list_new(
          tax_form_line_list );

list_set_list(
   latex_table_list,
   tax_form_account_latex_table_list->
      latex_table_list );

TAX_FORM_ENTITY_LATEX_LIST *
    tax_form_entity_latex_list =
       tax_form_entity_latex_list_new(
          tax_form_line_list );

list_set_list(
   latex_table_list,
   tax_form_entity_latex_table_list->
      latex_table_list );

/* Attributes */
STATEMENT_LINK *statement_link;
LATEX *latex;
LIST *latex_table_list;
TAX_FORM_LINE_LATEX_TABLE *
   tax_form_line_latex_table;
TAX_FORM_ACCOUNT_LATEX_LIST *
   tax_form_account_latex_list;
TAX_FORM_ENTITY_LATEX_LIST *
   tax_form_entity_latex_list;
```



TAX_FORM_LINE_LATEX_TABLE

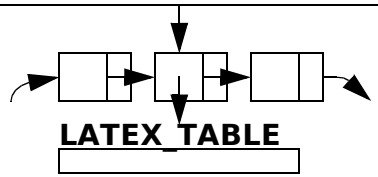
```
/* Usage */
TAX_FORM_LINE_LATEX_TABLE *
   tax_form_line_latex_table_new(
      char *tax_form_name,
      char *statement_caption_title,
      char *statement_caption_sub_title,
      LIST *tax_form_line_list );

/* Process */
TAX_FORM_LINE_LATEX_TABLE *
   tax_form_line_latex_table_calloc(
      void );

char *tax_form_line_latex_table_caption(
   char *tax_form_name,
   char *statement_caption_title,
   char *statement_caption_sub_title );

LIST *tax_form_line_latex_table_column_list();

LIST *tax_form_line_latex_table_row_list(
   tax_form_line_list,
   tax_form_line_latex_table_column_list() );

if ( !list_length( tax_form_line_latex_table_row_list() ) )
      return NULL;

LATEX_TABLE *latex_table =
   latex_table_new(
      tax_form_line_latex_table_caption() /* title */,
      tax_form_entity_latex_column_list(),
      tax_form_entity_latex_row_list() );

/* Usage */
LIST *tax_form_line_latex_table_column_list(
   void );

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

list_set(
   latex_column_list,
   LATEX_COLUMN *latex_column_new(
      "tax_form_line",
      latex_column_text /* latex_column_enum */,
      0 /* float_decimal_count */,
      (char *)0 /* paragraph_size */,
      1 /* first_column_boolean */ ) );

list_set(
   latex_column_list,
   LATEX_COLUMN *latex_column_new(
      "tax_form_description",
      latex_column_text /* latex_column_enum */,
      0 /* float_decimal_count */,
      (char *)0 /* paragraph_size */,
      0 /* not first_column_boolean */ ) );

list_set(
   latex_column_list,
   latex_column_new(
      "balance",
      latex_column_float /* latex_column_enum */,
      0 /* float_decimal_count */,
      (char *)0 /* paragraph_size */,
      0 /* not first_column_boolean */ ) );

/* Usage */
LIST *tax_form_line_latex_table_row_list(
   LIST *tax_form_line_list,
   LIST *tax_form_line_latex_table_column_list() );

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

for TAX_FORM_LINE *tax_form_line in tax_form_line_list
{
   if ( !tax_form_line->total ) continue;

   list_set(
      row_list,
         LATEX_ROW *tax_form_line_latex_table_row(
            tax_form_line,
            tax_form_line_latex_table_column_list ) );
   }
}

/* Usage */
LATEX_ROW *tax_form_line_latex_table_row(
   TAX_FORM_LINE *tax_form_line,
   LIST *latex_column_list );

/* Process */
list_rewind( latex_column_list );

LIST *latex_cell_list = list_new();

LATEX_COLUMN *latex_column =
   list_get( latex_column_list );
list_next( latex_column_list );

list_set(
   latex_cell_list,
   LATEX_CELL *latex_cell_small_new(
      latex_column,
      0 /* not first_row_boolean */,
      tax_form_line->tax_form_line_string ) );

LATEX_COLUMN *latex_column =
   list_get( latex_column_list );
list_next( latex_column_list );

list_set(
   latex_cell_list,
   LATEX_CELL *latex_cell_small_new(
      latex_column,
      0 /* not first_row_boolean */,
      tax_form_line->description ) );

LATEX_COLUMN *latex_column =
   list_get( latex_column_list );
list_next( latex_column_list );

list_set(
   latex_cell_list,
   LATEX_CELL *latex_cell_float_new(
      latex_column,
      0 /* not first_row_boolean */,
      tax_form_line->total /* value */ ) );

LATEX_ROW *latex_row_new( latex_cell_list );

/* Attribute */
LATEX_TABLE *latex_table;
```

TAX_FORM_ACCOUNT_LATEX_LIST

```
/* Usage */
TAX_FORM_ACCOUNT_LATEX_LIST *
    tax_form_account_latex_list_new(
        LIST *tax_form_line_list );

/* Process */
TAX_FORM_ACCOUNT_LATEX_LIST *
    tax_form_account_latex_list_calloc(
        void );
```

```
LIST *latex_table_list = list_new();

for TAX_FORM_LINE *tax_form_line in tax_form_line_list
{
    if ( !tax_form_line->total ) continue;

    list_set(
        latex_table_list,
        LATEX_TABLE *tax_form_account_latex_table(
            tax_form_line->string,
            tax_form_line->
                tax_form_line_account_list,
            tax_form_line->description,
            tax_form_line->total ) );
}
```

```
/* Attribute */
LIST *latex_table_list;
```

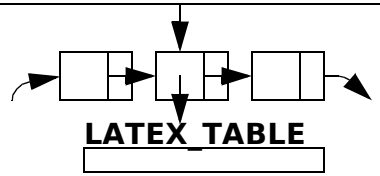


TAX_FORM_ACCOUNT_LATEX

```
/* Usage */
LATEX_TABLE *tax_form_account_latex_table(
    char *tax_form_line_string,
    LIST *tax_form_line_account_list,
    char *tax_form_line_description,
    double tax_form_line_total );

/* Process */
char *tax_form_account_latex_caption(
    char *tax_form_line_string,
    char *tax_form_line_description,
    double tax_form_line_total );

LIST *tax_form_account_latex_column_list();

LIST *tax_form_account_latex_row_list(
    tax_form_line_account_list,
    tax_form_account_latex_column_list() );

LATEX_TABLE *latex_table_new(
    tax_form_account_latex_caption() /* title */,
    tax_form_account_latex_column_list(),
    tax_form_account_latex_row_list() );

/* Usage */
LIST *tax_form_account_latex_column_list(
    void );

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

list_set(
    latex_column_list,
    latex_column_new(
       “account_name“,
       latex_column_text /* latex_column_enum */,
       0 /* float_decimal_count */,
       (char *)0 /* paragraph_size */,
       1 /* first_column_boolean */ ) );

list_set(
    column_list,
    latex_column_new(
       “total”,
       latex_column_float /* latex_column_enum */,
       0 /* float_decimal_count */,
       (char *)0 /* paragraph_size */,
       0 /* not first_column_boolean */ ) );

/* Usage */
LIST *tax_form_account_latex_row_list(
    LIST *tax_form_line_account_list,
    LIST *tax_form_account_latex_column_list() );

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

for TAX_FORM_LINE_ACCOUNT *
       tax_form_line_account in
          tax_form_line_account_list
{
    if ( !tax_form_line_account->journal_list_total )
    {
       continue;
    }

    list_set(
       latex_row_list,
       tax_form_line_account_latex_row(
          tax_form_line_account,
          tax_form_account_latex_column_list ) );
}

/* Usage */
LATEX_ROW *tax_form_line_account_latex_row(
    TAX_FORM_LINE_ACCOUNT *tax_form_line_account,
    LIST *latex_column_list );

/* Process */
list_rewind( latex_column_list );

LIST *latex_cell_list = list_new();

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
list_next( latex_column_list );

char *string_initial_capital(
    buffer[],
    tax_form_line_account->account_name );

list_set(
    latex_cell_list,
    LATEX_CELL *latex_cell_small_new(
       latex_column,
       0 /* not first_row_boolean */,
       strdup( buffer ) ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
list_next( latex_column_list );

list_set(
    latex_cell_list,
    LATEX_CELL *latex_cell_float_new(
       latex_column,
       0 /* not first_row_boolean */,
       tax_form_line_account->
          journal_list_total /* value */ ) );

LATEX_ROW *latex_row_new( latex_cell_list );
```

TAX_FORM_ENTITY_LATEX_LIST

```
/* Usage */
TAX_FORM_ENTITY_LATEX_LIST *
    tax_form_entity_latex_list_new(
       LIST *tax_form_line_list );

/* Process */
TAX_FORM_ENTITY_LATEX_LIST *
    tax_form_entity_latex_list_calloc(
       void );

LIST *latex_table_list = list_new();

for TAX_FORM_LINE *tax_form_line in tax_form_line_list
{
    if ( !tax_form_line->total ) continue;

for TAX_FORM_LINE_ACCOUNT *
    tax_form_line_account in
       tax_form_line->
          tax_form_line_account_list
{
    if ( !tax_form_line_account->journal_list_total )
    {
       continue;
    }

    LATEX_TABLE *tax_form_entity_latex_table(
       tax_form_line->string,
       tax_form_line_account->
          account_name,
          tax_form_line_account->
             journal_list_total,
          tax_form_line_account->
             tax_form_entity_list );

       list_set(
          latex_table_list,
          tax_form_entity_latex_table() );
    }
}

/* Attribute */
LIST *latex_table_list;
```



TAX_FORM_ENTITY_LATEX

```
/* Usage */
LATEX_TABLE *tax_form_entity_latex_table(
    char *tax_form_line_string,
    char *account_name,
    double journal_list_total,
    LIST *tax_form_entity_list );

/* Process */
LIST *tax_form_entity_latex_column_list();

LIST *tax_form_entity_latex_row_list(
     tax_form_entity_list,
     tax_form_entity_latex_column_list() );

char *tax_form_entity_latex_caption(
    char *tax_form_line_string,
    char *account_name,
    double journal_list_total );

char *date_now16( date_utc_offset() );

LATEX_TABLE *latex_table_new(
    tax_form_entity_latex_caption() /* title */,
    tax_form_entity_latex_column_list(),
    tax_form_entity_latex_row_list() );

/* Usage */
LIST *tax_form_entity_latex_column_list(
    void );

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

list_set(
    column_list,
       latex_column_new(
          “full_name“,
          latex_column_text /* latex_column_enum */,
          0 /* float_decimal_count */,
          (char *)0 /* paragraph_size */,
          1 /* first_column_boolean */ ) );

list_set(
    column_list,
    latex_column_new(
       “total”,
       latex_column_float /* latex_column_enum */,
       0 /* float_decimal_count */,
       (char *)0 /* paragraph_size */,
       0 /* not first_column_boolean */ ) );

/* Usage */
LIST *tax_form_entity_latex_row_list(
     LIST *tax_form_entity_list,
     LIST *latex_column_list );

/* Process */
for TAX_FORM_ENTITY *
       tax_form_entity in
          tax_form_entity_list
{
    if ( !tax_form_entity->total ) continue;

    LATEX_ROW *tax_form_entity_latex_row(
       tax_form_entity,
       latex_column_list );

    list_set(
       latex_row_list,
       tax_form_entity_latex_row() );
}

/* Usage */
LATEX_ROW *tax_form_entity_latex_row(
    TAX_FORM_ENTITY *tax_form_entity,
    LIST *latex_column_list );

/* Process */
list_rewind( latex_column_list );

LIST *cell_list = list_new();

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
list_next( latex_column_list );

list_set(
    cell_list,
    latex_cell_small_new(
       latex_column,
       0 /* not first_row_boolean */,
       tax_form_entity->entity->full_name ) );

LATEX_COLUMN *latex_column =
    list_get( latex_column_list );
list_next( latex_column_list );

list_set(
    cell_list,
    latex_cell_float_new(
       latex_column,
       0 /* not first_row_boolean */,
       tax_form_entity->total /* value */ ) );

LATEX_ROW *latex_row_new( cell_list );
```

CLOSE_NOMINAL (1)

```
/* Usage */
CLOSE_NOMINAL *close_nominal_fetch(
    char *application_name,
    char *process_name,
    char *as_of_date_string,
    boolean undo );

/* Process */
CLOSE_NOMINAL *close_nominal_calloc( void );

if ( undo )
{
    CLOSE_NOMINAL_UNDO *close_nominal_undo =
       close_nominal_undo_fetch(
          void );

    if ( !close_nominal_undo->
          transaction_date_close_nominal_undo->
          transaction_date_time_memo_maximum_string )
    {
       char *
       close_nominal_undo_no_transaction_message(
          void );

       as_of_date_string = (char *)0;

       goto set_statement_caption;
    }

    as_of_date_string =
       close_nominal_undo->
       transaction_date_close_nominal_undo->
       transaction_date_time_memo_maximum_string;
}
else
{
    boolean transaction_date_as_of_date_populated(
       as_of_date_string );

    if ( !transaction_date_as_of_date_populated() )
    {
       char *close_nominal_do_empty_date_message(
          void );

       as_of_date = (char *)0;

       goto set_statement_caption;
    }

    CLOSE_NOMINAL_DO *close_nominal_do =
       close_nominal_do_fetch(
          as_of_date_string );

    if ( close_nominal_do->
          transaction_date_close_nominal_do->
          transaction_date_close_boolean )
    {
       char *
       close_nominal_do_transaction_exists_message(
          close_nominal_do->
             transaction_date_close_nominal_do->
             transaction_date_close_date_time );

       goto set_statement_caption;
    }

    if ( close_nominal_do->no_transactions_boolean )
    {
       char *close_nominal_do_no_transaction_message(
          void );
    }
}

set_statement_caption:

STATEMENT_CAPTION *statement_caption =
    statement_caption_new(
       application_name,
       process_name,
       (char *)0 /* begin_date_string */,
       as_of_date_string /* end_date_string */ );

/* Attributes */
CLOSE_NOMINAL_DO *close_nominal_do;
CLOSE_NOMINAL_UNDO *close_nominal_undo;
char *undo_no_transaction_message;
char *empty_date_message;
char *do_transaction_exists_message;
char *do_no_transaction_message;
STATEMENT_CAPTION *statement_caption;
```

CLOSE_NOMINAL (2)

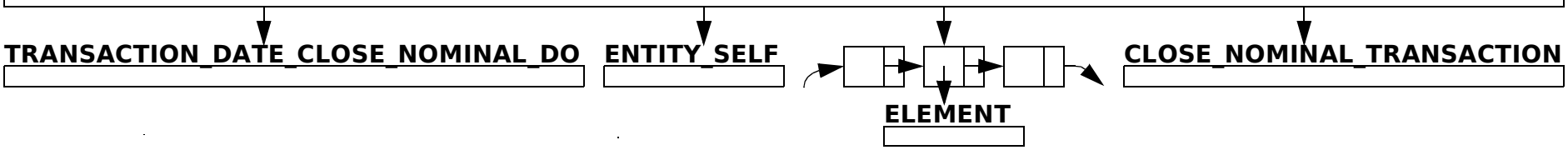
```
/* Driver */
document_process_output(
    application_name,
    (LIST *)0 /* javascript_filename_list */,
    (char *)0 /* title */ );

printf(
    "%s\n",
    close_nominal->
       statement_caption->
       frame_title );

if ( close_nominal->
       undo_no_transaction_message )
{
    printf( "%s\n",
       close_nominal->
          undo_no_transaction_message );
}
else
if ( close_nominal->
       do_no_transaction_message )
{
    printf( "%s\n",
       close_nominal->
          do_no_transaction_message );
}
else
if ( close_nominal->empty_date_message )
{
    printf( "%s\n",
       close_nominal->
          empty_date_message );
}
else
if ( close_nominal->do_transaction_exists_message )
{
    printf( "%s\n",
       close_nominal->
          do_transaction_exists_message );
}
else
if ( close_nominal->close_nominal_do )
{
    close_nominal_accounts_do(
       CLOSE_NOMINAL_DO *
          close_nominal->close_nominal_do,
       boolean execute );

    if ( execute )
    {
       printf( "%s\n",
          char *
          close_nominal_do_execute_message(
             char *close_nominal->
                close_nominal_do->
                transaction_date_close_nominal_do->
                transaction_date_close_date_time ) );
    }
    else
    {
       printf( "%s\n",
          char *
          close_nominal_do_no_execute_message(
             void ) );
    }
}
else
{
    if ( !execute )
    {
       printf( "%s\n",
          char *
          close_nominal_undo_no_execute_message(
             void ) );
    }

    close_nominal_accounts_undo(
       CLOSE_NOMINAL_UNDO *
          close_nominal->close_nominal_undo,
       boolean execute );

    if ( execute )
    {
       printf( "%s\n",
          char *
          close_nominal_undo_execute_message(
             void ) );
    }
}

document_close();
```

CLOSE_NOMINAL_DO (1)

```
/* Usage */
CLOSE_NOMINAL_DO *close_nominal_do_fetch(
    char *as_of_date_string );

/* Process */
CLOSE_NOMINAL_DO *close_nominal_do_calloc(
    void );

TRANSACTION_DATE_CLOSE_NOMINAL_DO *
    transaction_date_close_nominal_do =
        transaction_date_close_nominal_do_new(
            as_of_date_string );

if ( transaction_date_close_nominal_do->
        transaction_date_close_boolean )
{
    return this;
}

LIST *close_nominal_do_element_name_list(
    char *ELEMENT_REVENUE,
    char *ELEMENT_EXPENSE,
    char *ELEMENT_GAIN,
    char *ELEMENT_LOSS );

LIST *element_statement_list(
    close_nominal_do_element_name_list(),
    transaction_date_close_nominal_do->
        transaction_date_close_date_time,
    1 /* fetch_subclassification_list */,
    1 /* fetch_account_list */,
    1 /* fetch_journal_latest */,
    0 /* not fetch_transaction */ );

void element_list_sum_set(
    element_statement_list() );

ELEMENT *revenue_element =
    element_seek(
        ELEMENT_REVENUE,
        element_statement_list() );

double revenue_sum =
    element_sum(
        revenue_element );

ELEMENT *gain_element =
    element_seek(
        ELEMENT_GAIN,
        element_statement_list() );

double gain_sum =
     element_sum(
         gain_element );

char *account_drawing_string(
    ACCOUNT_DRAWING_KEY );

double close_nominal_do_drawing_sum(
    char *transaction_date_close_nominal_do->
        transaction_date_close_date_time,
    char *account_drawing_string() );

double close_nominal_do_debit_sum(
    double revenue_sum,
    double gain_sum,
    double close_nominal_do_drawing_sum() );

ELEMENT *expense_element =
    element_seek(
        ELEMENT_EXPENSE,
        element_statement_list() );

double expense_sum =
     element_sum(
         expense_element );

ELEMENT *loss_element =
    element_seek(
        ELEMENT_LOSS,
        element_statement_list() );

double loss_sum = element_sum( loss_element );

double close_nominal_do_credit_sum(
    double expense_sum,
    double loss_sum );

double close_nominal_do_transaction_amount(
    double close_nominal_do_debit_sum(),
    double close_nominal_do_credit_sum() );

boolean close_nominal_do_no_transactions_boolean(
    double close_nominal_do_transaction_amount() );

if ( close_nominal_do_no_transactions_boolean() )
{
    return this;
}

double close_nominal_do_retained_earnings(
    double revenue_sum,
    double gain_sum,
    double expense_sum,
    double loss_sum,
    double close_nominal_do_drawing_sum() );

char *account_closing_entry_string(
    ACCOUNT_CLOSING_KEY,
    ACCOUNT_EQUITY_KEY,
    __FUNCTION__ );

ENTITY_SELF *entity_self =
    entity_self_fetch(
        0 /* not fetch_entity_boolean */ );

CLOSE_NOMINAL_TRANSACTION *
    close_nominal_transaction =
        close_nominal_transaction_new(
            element_statement_list() /* in/out */,
            transaction_date_close_nominal_do->
                transaction_date_close_date_time,
            account_drawing(),
            close_nominal_do_drawing_sum(),
            close_nominal_do_transaction_amount(),
            close_nominal_do_retained_earnings(),
            account_closing_entry_string(),
            entity_self->full_name,
            entity_self->street_address );

double journal_debit_sum(
    close_nominal_do_transaction->
        transaction->
        journal_list );

double journal_credit_sum(
    close_nominal_do_transaction->
        transaction->
        journal_list );
```

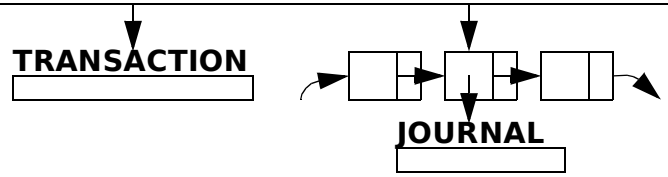
CLOSE_NOMINAL_DO (2)

```
/* Driver */
if ( execute )
{
   void transaction_insert(
      close_nominal_do->
         close_nominal_transaction->
         transaction->
         full_name,
      close_nominal_do->
         close_nominal_transaction->
         transaction->
         street_address,
      close_nominal_do->
         close_nominal_transaction->
         transaction->
         transaction_date_time,
      close_nominal_do->
         close_nominal_transaction->
         transaction->
         transaction_amount,
      close_nominal_do->
         close_nominal_transaction->
         transaction->
         check_number,
      close_nominal_do->
         close_nominal_transaction->
         transaction->
         memo,
      TRANSACTION_LOCK_Y,
      close_nominal_do->
         close_nominal_transaction->
         transaction->
         journal_list,
      1 /* insert_journal_list_boolean */ );
}
else
{
   void journal_list_sum_html_display(
      close_nominal_do->
         close_nominal_transaction->
      close_nominal_do->
         close_nominal_transaction->
         transaction->
         journal_list,
      close_nominal_do->
         close_nominal_transaction->
         transaction->
         transaction_date_time,
      close_nominal_do->
         close_nominal_transaction->
         transaction->
         transaction_memo
      close_nominal_do->
         journal_debit_sum,
      close_nominal_do->
         journal_credit_sum );
}

/* Attributes */
TRANSACTION_DATE_CLOSE_NOMINAL_DO *
   transaction_date_close_nominal_do;
LIST *element_name_list;
LIST *element_statement_list;
ELEMENT *revenue_element;
double revenue_sum;
ELEMENT *gain_element;
double gain_sum;
char *account_drawing_string;
double drawing_sum;
double debit_sum;
ELEMENT *expense_element;
double expense_sum;
ELEMENT *loss_element;
double loss_sum;
double credit_sum;
double transaction_amount;
boolean no_transactions_boolean;
double retained_earnings;
char *account_closing_entry_string;
ENTITY_SELF *entity_self;
CLOSE_NOMINAL_TRANSACTION *
   close_nominal_transaction;
double journal_debit_sum;
double journal_credit_sum;
```



CLOSE_NOMINAL_TRANSACTION (1)

```
/* Usage */
CLOSE_NOMINAL_TRANSACTION *
   close_nominal_transaction_new(
      LIST *element_statement_list() /* in/out */,
      char *transaction_date_close_nominal_do->
         transaction_date_close_date_time,
      char *account_drawing_string(),
      double close_nominal_do_drawing_sum(),
      double close_nominal_do_transaction_amount(),
      double close_nominal_do_retained_earnings(),
      char *account_closing_entry_string(),
      char *full_name,
      char *street_address );

/* Process */
CLOSE_NOMINAL_TRANSACTION *
   close_nominal_transaction_calloc(
      void );

TRANSACTION *transaction =
   transaction_new(
      full_name,
      street_address,
      transaction_date_close_date_time );

transaction->transaction_amount =
   close_nominal_do_transaction_amount;

transaction->memo =
   TRANSACTION_CLOSE_MEMO;

transaction->journal_list =
   LIST *close_nominal_transaction_journal_list(
      element_statement_list /* in/out */,
      transaction_date_close_date_time,
      account_drawing_string,
      close_nominal_do_drawing_sum,
      close_nominal_do_retained_earnings,
      account_closing_entry_string,
      full_name,
      street_address );

/* Usage */
LIST *close_nominal_transaction_journal_list(
   LIST *element_statement_list /* in/out */,
   char *transaction_date_close_date_time,
   char *account_drawing_string,
   double close_nominal_do_drawing_sum,
   double close_nominal_do_retained_earnings,
   char *account_closing_entry_string,
   char *full_name,
   char *street_address );

/* Process */
void element_list_account_statement_list_set(
   element_statement_list );

LIST *journal_list = list_new();

for ELEMENT *element in element_statement_list
{
   if ( !element->sum ) continue;

   for ACCOUNT *account in
      element->account_statement_list
   {
      if ( !account->account_journal_latest->balance )
            continue;

      JOURNAL *journal =
         journal_new(
            full_name,
            street_address,
            transaction_date_close_date_time,
            account->account_name );

      ACCOUNT *journal_account =
         ACCOUNT *account_fetch(
            journal->account_name,
            1 /* fetch_subclassification */,
            1 /* fetch_element */ );

      journal->debit_amount =
         close_nominal_transaction_debit_amount(
            element->accumulate_debit,
            account->
               account_journal_latest->
               balance );

      journal->credit_amount =
         close_nominal_transaction_credit_amount(
            element->accumulate_debit,
            account->
               account_journal_latest->
               balance );

      list_set(
         journal_list,
         journal );
   }
}

if ( close_nominal_do_drawing_sum )
{
   list_set(
      journal_list,
      JOURNAL *
         close_nominal_transaction_drawing_journal(
            transaction_date_close_date_time,
            account_drawing_string,
            close_nominal_do_drawing_sum,
            full_name,
            street_address ) );
}

if ( close_nominal_do_retained_earnings )
{
   list_set(
      journal_list,
      JOURNAL *
         close_nominal_transaction_close_journal(
            transaction_date_close_date_time,
            close_nominal_do_retained_earnings,
            account_closing_entry_string,
            full_name,
            street_address ) );
}

/* Attribute */
TRANSACTION *transaction;
```



CLOSE_NOMINAL_TRANSACTION (2)

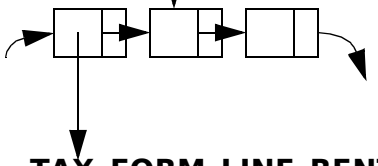
```
/* Usage */
double close_nominal_transaction_debit_amount(
    boolean element_accumulate_debit,
    double balance );

/* Process */
if ( !element_accumulate_debit )
    debit_amount = balance;
else
    debit_amount = 0.0;

/* Usage */
double close_nominal_transaction_credit_amount(
    boolean element_accumulate_debit,
    double balance );

/* Process */
if ( element_accumulate_debit )
    credit_amount = balance;
else
    credit_amount = 0.0;

/* Usage */
JOURNAL *close_nominal_transaction_drawing_journal(
    char *transaction_date_time_closing,
    char *account_drawing_string,
    double close_nominal_do_drawing_sum,
    char *full_name,
    char *street_address );

/* Process */
JOURNAL *journal =
    JOURNAL *journal_new(
        full_name,
        street_address,
        transaction_date_time_closing,
        account_drawing_string );

ACCOUNT *journal->account =
    ACCOUNT *account_fetch(
        journal->account_name,
        1 /* fetch_subclassification */,
        1 /* fetch_element */ );

journal->debit_amount =
    close_nominal_do_drawing_sum;

/* Usage */
JOURNAL *close_nominal_transaction_close_journal(
    char *transaction_date_time_closing,
    double close_nominal_do_retained_earnings,
    char *account_closing_entry_string,
    char *full_name,
    char *street_address );

/* Process */
JOURNAL *journal =
    JOURNAL *journal_new(
        full_name,
        street_address,
        transaction_date_time_closing,
        account_closing_entry_string );

ACCOUNT *journal->account =
    ACCOUNT *account_fetch(
        journal->account_name,
        1 /* fetch_subclassification */,
        1 /* fetch_element */ );

if ( close_nominal_do_retained_earnings > 0.0 )
{
    journal->credit_amount =
        close_nominal_do_retained_earnings;
}
else
{
    journal->debit_amount =
        -close_nominal_do_retained_earnings;
}
```

CLOSE_NOMINAL_UNDO

```
/* Usage */
CLOSE_NOMINAL_UNDO *close_nominal_undo_fetch(
    void );

/* Process */
CLOSE_NOMINAL_UNDO *close_nominal_undo_calloc(
    void );

ENTITY_SELF *entity_self =
    ENTITY_SELF *entity_self_fetch(
        0 /* not fetch_entity_boolean */ );

TRANSACTION_DATE_CLOSE_NOMINAL_UNDO *
    transaction_date_close_nominal_undo =
        transaction_date_close_nominal_undo_new(
            entity_self->full_name,
            entity_self->street_address );

if ( !transaction_date_close_nominal_undo->
        transaction_date_time_memo_maximum_string )
{
    return this;
}

TRANSACTION *transaction =
    TRANSACTION *transaction_fetch(
        entity_self->full_name,
        entity_self->street_address,
        transaction_date_close_nominal_undo->
            transaction_date_time_memo_maximum_string,
        1 /* fetch_journal_list */ );

/* Driver */
void close_nominal_undo_display(
    TRANSACTION *
        close_nominal_undo->
        transaction );

/* Process */
void transaction_html_display( transaction );

/* Driver */
void close_nominal_undo_execute(
    TRANSACTION *
        close_nominal_undo->
        transaction );

/* Process */
void transaction_delete(
    transaction->full_name,
    transaction->street_address,
    transaction->transaction_date_time );

/* Driver */
char *close_nominal_undo_execute_message(
    char *close_nominal->
        close_nominal_undo->
        transaction->
        transaction_date_time );

/* Process */

/* Attributes */
ENTITY_SELF *entity_self;
TRANSACTION_DATE_CLOSE_NOMINAL_UNDO *
    transaction_date_close_nominal_undo;
TRANSACTION *transaction;
```

TAX_RENTAL_PROPERTY

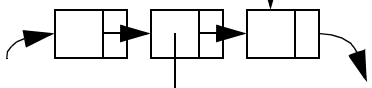
```
/* Usage */
LIST *tax_rental_property_list_fetch(
    char *tax_form_string,
    int tax_year );

/* Attributes */
char  *property_street_address;

/* Process */
LIST *tax_rental_property_string_list(
    tax_rental_property_list );

LIST *tax_form_line_rental_list(
    tax_form_line_list(),
    tax_rental_property_string_list );

double recovery_amount;
double revenue_total [23a];
double mortgage_interest_total [23c];
double depreciation_total [23d];
double expense_total [23e];
double positive_net_income [24];
double loss_total [25];
double tax_rental_property_total();
```



TAX_FORM_LINE_RENTAL

```
char *tax_form_line;
char *tax_form_description;

enum tax_form_line_category;
LIST *tax_form_line_account_list();

LIST *rental_property_list;
double tax_form_line_rental_property_total();
```

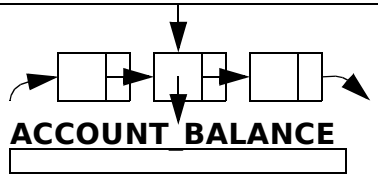
revenue
mortgage_interest_paid
depreciation_expense
operating_expense
loss



TAX_FORM_LINE_ACCOUNT

INVESTMENT_ACCOUNT (1)

```
/* Usage */
INVESTMENT_ACCOUNT *investment_account_fetch(
    char *full_name,
    char *street_address,
    char *account_number,
    boolean fetch_account_balance_list );

/* Process */
char *investment_account_primary_where(
    full_name,
    street_address,
    account_number );

char *appaserver_system_string(
    INVESTMENT_ACCOUNT_SELECT,
    INVESTMENT_ACCOUNT_TABLE,
    investment_account_primary_where() );

char *string_pipe_input(
    appaserver_system_string() );

if ( !string_pipe_input() ) return NULL;

INVESTMENT_ACCOUNT *investment_account_parse(
    full_name,
    street_address,
    account_number,
    investment_account_primary_where(),
    string_pipe_input() );

if ( fetch_account_balance_list )
{
    LIST *account_balance_list(
        full_name,
        street_address,
        account_number,
        investment_account_primary_where() );

    LIST *account_balance_update_set(
        account_balance_list() /* in/out */ );

    double investment_account_update_balance_latest(
        LIST *account_balance_update_set() );
}

/* Usage */
INVESTMENT_ACCOUNT *investment_account_parse(
    char *full_name,
    char *street_address,
    char *account_number,
    char *investment_account_primary_where(),
    char *input );

/* Process */
INVESTMENT_ACCOUNT *investment_account_new(
    full_name,
    street_address,
    account_number,
    investment_account_primary_where );

/* Usage */
INVESTMENT_ACCOUNT *investment_account_new(
    char *full_name,
    char *street_address,
    char *account_number,
    char *investment_account_primary_where() );

/* Process */
INVESTMENT_ACCOUNT *investment_account_calloc(
    void );

/* Attributes */
char *full_name,
char *street_address,
char *account_number;
char *primary_where;
char *investment_classification;
char *investment_purpose;
char *investment_classification;
char *investment_purpose;
int certificate_maturity_months;
char *certificate_maturity_date;
double interest_rate;
double balance_latest;
LIST *account_balance_list;
double update_balance_latest;
```



INVESTMENT_ACCOUNT (2)

```
/* Usage */
char *investment_account_primary_where(
    char *full_name,
    char *street_address,
    char *account_number );

/* Process */

/* Driver */

/* Returns error_string or null */
char *investment_account_update(
    const char *INVESTMENT_ACCOUNT_TABLE,
    char *investment_account->primary_where,
    double investment_account->update_balance_latest
);

/* Process */
char *investment_account_update_statement(
    const char *INVESTMENT_ACCOUNT_TABLE,
    char *primary_where,
    double update_balance_latest );

SPOOL *spool_new(
    "sql.e 2>&1",
    0 /* not capture_stderr_boolean */ );

fprintf(
    spool_new()->output_pipe,
    "%s\n",
    investment_account_update_statement() );

pclose( spool_new()->output_pipe );

LIST *list = spool_list( spool_new()->output_filename );

char *error_string = list_first( list );
```

ACCOUNT_BALANCE (1)

```
/* Usage */
LIST *account_balance_list(
    char *full_name,
    char *street_address,
    char *account_number,
    char *investment_account_primary_where() );

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

char *account_balance_system_string(
    ACCOUNT_BALANCE_SELECT,
    ACCOUNT_BALANCE_TABLE,
    investment_account_primary_where );

FILE *appaserver_input_pipe(
    account_balance_system_string() );

for char *string_input( input[], appaserver_input_pipe() )
{
    ACCOUNT_BALANCE *account_balance_parse(
        full_name,
        street_address,
        account_number,
        string_input() );

    list_set(
        list,
        account_balance_parse() );
}
pclose( appaserver_input_pipe() );

/* Usage */
char *account_balance_system_string(
    const char *select,
    const char *ACCOUNT_BALANCE_TABLE,
    char *where );

/* Process */
sprintf(
    system_string[],
    "select.sh \"%s\" %s \"%s\" date",
    select,
    ACCOUNT_BALANCE_TABLE,
    where );

/* Usage */
ACCOUNT_BALANCE *account_balance_parse(
    char *full_name,
    char *street_address,
    char *account_number,
    char *input );

/* Process */
ACCOUNT_BALANCE *account_balance_new(
    full_name,
    street_address,
    account_number,
    strdup( date_string[] ) );

/* Usage */
ACCOUNT_BALANCE *account_balance_new(
    char *full_name,
    char *street_address,
    char *account_number,
    char *date_string );

/* Process */
ACCOUNT_BALANCE *account_balance_calloc(
    void );

/* Usage */
LIST *account_balance_update_set(
    LIST *account_balance_list() /* in/out */ );

/* Process */
for ACCOUNT_BALANCE *account_balance in
    account_balance_list
{
    if ( list_first_boolean( account_balance_list ) )
    {
        account_balance->
            update_balance_change =
                account_balance->balance;

        account_balance->
            update_balance_change_percent = 100;
    }
    else
    {
        double account_balance_update_change(
            account_balance_prior->balance,
            account_balance->balance );

        account_balance->
            update_balance_change =
                account_balance_update_change();

        int account_balance_update_change_percent(
            account_balance_prior->balance,
            account_balance_update_change() );

        account_balance->
            update_balance_change_percent =
                account_balance_update_change_percent();
    }

    ACCOUNT_BALANCE *account_balance_prior =
        account_balance;
}

/* Attributes */
char *full_name;
char *street_address;
char *account_number;
char *date_string;
double balance;
double balance_change;
int balance_change_percent;

/* Set externally */
double update_balance_change;
int update_balance_change_percent;
```

ACCOUNT_BALANCE (2)

```
/* Usage */
double account_balance_update_change(
    double account_balance_prior->balance
       /* prior_balance */,
    double account_balance->balance );

/* Process */
return balance - prior_balance;

/* Usage */
int account_balance_update_change_percent(
    double account_balance_prior->balance
       /* prior_balance */,
    double account_balance_update_change()
       /* balance_change */ );

/* Process */
if ( !prior_balance && !balance_change )
    return 0;
else
if ( !prior_balance && balance_change < 0.0 )
    return -100;
else
if ( !prior_balance && balance_change > 0.0 )
    return 100;

double change_ratio =
    balance_change /
    prior_balance;

int change_percent =
    float_round_int(
       change_ratio * 100.0 );

if (    change_percent == -100
&&   !float_virtually_same(
          prior_balance,
          -balance_change ) )
{
    change_percent = -99;
}

/* Driver */

/* Returns error_string */
char *account_balance_list_update(
    LIST *investment_account->
       account_balance_list );

/* Process */
char *account_balance_update_system_string(
    ACCOUNT_BALANCE_TABLE,
    ACCOUNT_BALANCE_PRIMARY_KEY );

SPOOL *spool_new(
    account_balance_update_system_string(),
    0 /* not capture_stderr_boolean */ );

for ACCOUNT_BALANCE *account_balance in
    account_balance_list
{
    char *account_balance_update_change_string(
       char *account_balance->full_name,
       char *account_balance->street_address,
       char *account_balance->account_number,
       char *account_balance->date_string,
       double account_balance->
          update_balance_change );

    fprintf(
       spool_new()->output_pipe,
       "%s\n",
       account_balance_update_change_string() );

    char *account_balance_update_percent_string(
       char *account_balance->full_name,
       char *account_balance->street_address,
       char *account_balance->account_number,
       char *account_balance->date_string,
       int account_balance->
          update_balance_change_percent );

    fprintf(
       spool_new()->output_pipe,
       "%s\n",
       account_balance_update_percent_string() );
}

pclose( spool_new()->output_pipe );

LIST *list = spool_list( spool_new()->output_filename );

char *error_string =
    list_string_delimited(
       list, "<br>" );

/* Usage */
char *account_balance_update_system_string(
    const char *ACCOUNT_BALANCE_TABLE,
    const char *ACCOUNT_BALANCE_PRIMARY_KEY );

/* Process */
sprintf(
    system_string[],
    "update_statement.e table=%s key=%s carrot=y |"
    "sql.e 2>&1",
    ACCOUNT_BALANCE_TABLE,
    ACCOUNT_BALANCE_PRIMARY_KEY );
```

INVESTMENT_TRANSACTION (1)

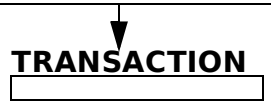
```
/* Usage */
INVESTMENT_TRANSACITON *
   investment_transaction_new(
      double investment_account_sum );

/* Process */
INVESTMENT_TRANSACTION *
   investment_transaction_calloc(
      void );

char *account_key_account_name(
   INVESTMENT_ACCOUNT_KEY,
   __FUNCTION__ );

char *date_now19(
   date_utc_offset() );

ACCOUNT_JOURNAL *
   account_journal_latest(
      JOURNAL_TABLE,
      account_key_account_name(),
      date_now19() /* end_date_time_string */,
      0 /* not fetch_transaction */ );

double investment_transaction_current_balance(
   ACCOUNT_JOURNAL *account_journal_latest() );

double investment_transaction_delta(
   double investment_account_sum,
   double investment_transaction_current_balance() );

boolean float_virtually_same(
   investment_transaction_delta(),
   0.0 );

if ( float_virtually_same ) return this;

ENTITY_SELF *entity_self_fetch(
   0 /* not fetch_entity_boolean */ );

if ( investment_transaction_delta() < 0.0 )
{
   TRANSACTION *transaction =
      investment_negative_transaction(
         account_key_account_name()
            /* investment_account_name */,
         date_now19()
            /* transaction_date_time */,
         -investment_transaction_delta()
            /* transaction_amount */,
         entity_self_fetch() );
}
else
if ( investment_transaction_delta() > 0.0 )
{
   TRANSACTION *transaction =
      investment_positive_transaction(
         account_key_account_name()
            /* investment_account_name */,
         date_now19()
            /* transaction_date_time */,
         investment_transaction_delta()
            /* transaction_amount */,
         entity_self_fetch() );
}

/* Usage */
TRANSACTION *investment_negative_transaction(
   char *investment_account_name,
   char *transaction_date_time,
   double transaction_amount,
   ENTITY_SELF *entity_self );

/* Process */
char *account_key_account_name(
   INVESTMENT_LOSS_ACCOUNT_KEY,
   __FUNCTION__ );

TRANSACTION *transaction_binary(
   entity_self->full_name,
   entity_self->street_address,
   transaction_date_time,
   transaction_amount,
   INVESTMENT_TRANSACTION_MEMO,
   account_key_account_name()
      /* debit_account_name */,
   investment_account_name
      /* credit_account_name */ );

/* Usage */
TRANSACTION *investment_positive_transaction(
   char *investment_account_name,
   char *transaction_date_time,
   double transaction_amount,
   ENTITY_SELF *entity_self );

/* Process */
char *account_key_account_name(
   INVESTMENT_GAIN_ACCOUNT_KEY,
   __FUNCTION__ );

TRANSACTION *transaction_binary(
   entity_self->full_name,
   entity_self->street_address,
   transaction_date_time,
   transaction_amount,
   INVESTMENT_TRANSACTION_MEMO,
   investment_account_name
      /* debit_account_name */,
   account_key_account_name()
      /* credit_account_name */ );

/* Attributes */
char *account_key_account_name;
char *date_now19;
ACCOUNT_JOURNAL *account_journal_latest;
double current_balance;
```

INVESTMENT_TRANSACTION (2)

```
/* Driver */
if ( !investment_transaction->transaction )
{
    printf(
        "%s\n",
        INVESTMENT_TRANSACTION_NO_CHANGE );

    exit( 0 );
}

if ( execute_boolean )
{
    char *investment_transaction->
        transaction->
transaction_date_time =
    transaction_insert(
        investment_transaction->
            transaction->
            full_name,
        investment_transaction->
            transaction->
            street_address,
        investment_transaction->
            transaction->
            transaction_date_time,
        investment_transaction->
            transaction->
            transaction_amount,
            0 /* check_number */,
        investment_transaction->
            transaction->
            memo,
        'n' /* lock_transaction_yn not there */,
        investment_transaction->
            transaction->
            journal_list,
        1 /* insert_journal_list_boolean */ );
}

void transaction_html_display(
    investment_transaction->transaction );
```

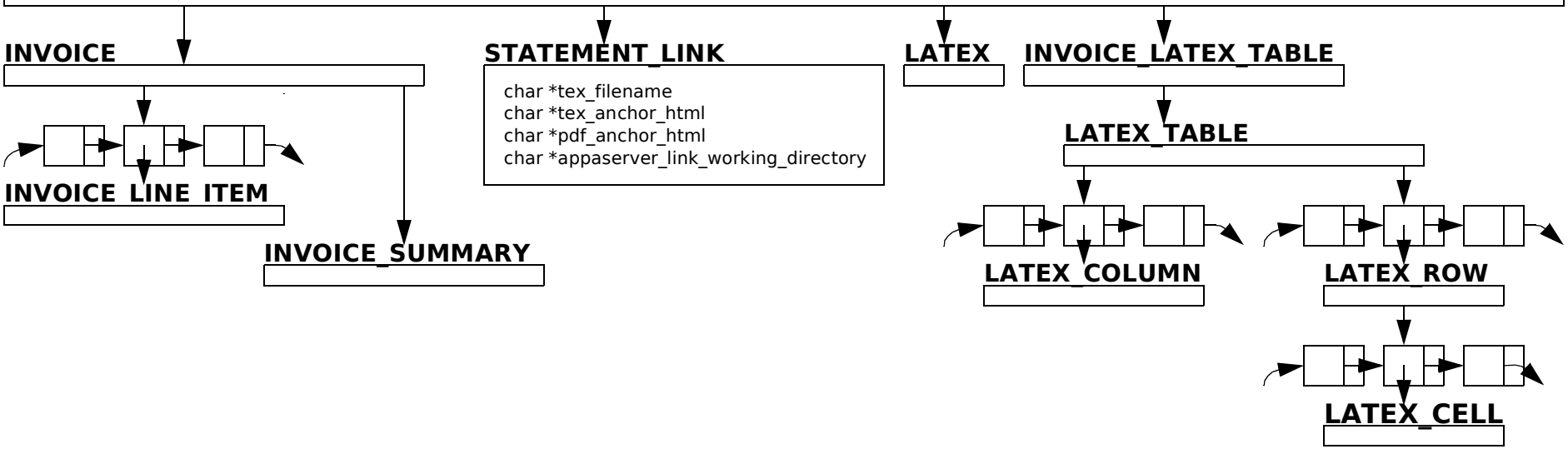
ACCOUNT_BALANCE_EQUITY

/* Attributes */	double book_value_balance
char *date_time	double moving_share_price
char *investment_operation	double unrealized_gain_balance
double share_price	double unrealized_gain_change
doubel share_quantity_change	double reaqlized_gain
double share_quantity_balance	boolean table_first_row
double cash_in	char *transaction_date_time;
double market_value	TRANSACTION *transaction;
double book_value_change	



INVOICE_LATEX (1)

```
/* Usage */
INVOICE_LATEX *invoice_latex_new(
    char *application_name,
    char *process_name,
    char *appaserver_parameter_data_directory,
    INVOICE *invoice );

/* Process */
INVOICE_LATEX *invoice_latex_calloc(
    void );

char *invoice_latex_document_header();

STATEMENT_LINK *statement_link =
    statement_link_new(
        application_name,
        process_name,
        appaserver_parameter_data_directory,
        (char *)0 /* begin_date_string */,
        (char *)0 /* end_date_string */,
        getpid() /* process_id */ );

char *statement_caption_logo_filename(
    STATEMENT_LOGO_FILENAME_KEY );

LATEX *latex =
    latex_new(
        statement_link->tex_filename,
        statement_link->
            appaserver_link_working_directory,
        0 /* not statement_pdf_landscape_boolean */,
        statement_caption_logo_filename() );

char *invoice_latex_title(
    invoice->title,
    statement_caption_logo_filename(),
    invoice->date_string );

INVOICE_LATEX_TABLE *invoice_latex_table =
    invoice_latex_table_new(
        invoice->invoice_enum,
        invoice->title,
        invoice->entity_self,
        invoice->customer,
        invoice->invoice_line_item_list,
        invoice->amount_due_label,
        invoice->invoice_summary );

char *invoice_latex_footer(
    enum invoice_enum invoice->invoice_enum );

/* Attributes */
INVOICE *Invoice;
char *document_header;
STATEMENT_LINK *statement_link;
char *statement_caption_logo_filename;
LATEX *latex;
char *title;
INVOICE_LATEX_TABLE *invoice_latex_table;
char *footer;
```



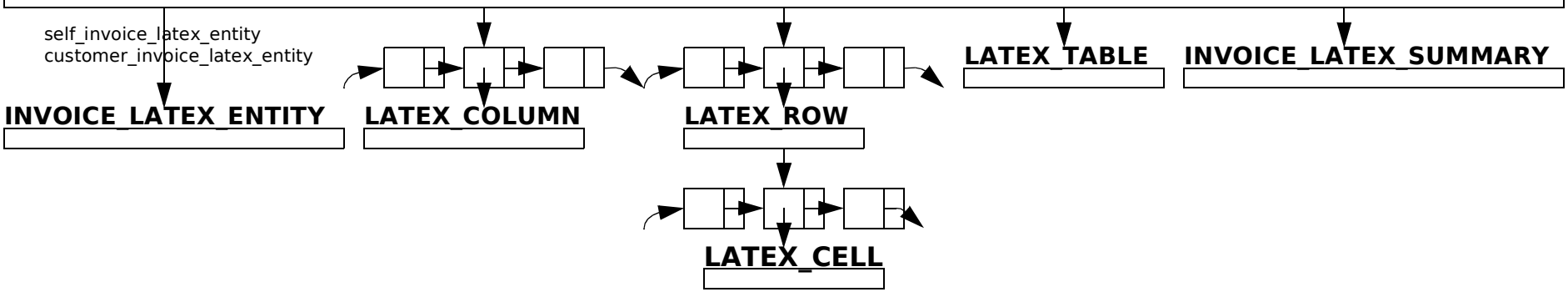
INVOICE_LATEX (2)

```
/* Usage */
char *invoice_latex_document_header(
    void );

/* Process */
return
    "\\documentclass{letter}\n"
    "\\usepackage{longtable}\n"
    "\\usepackage{graphics}\n"
    "\\usepackage[ margin=1in,\n"
    "                    nohead]{geometry}\n"
    "\\begin{document}\n";

/* Usage */
char *invoice_latex_title(
    char *invoice_title,
    char *invoice_latex->
       statement_caption_logo_filename,
    char *invoice->date_string );

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

ptr + = sprintf( ptr,
“\\begin{center}{\\Large \\bf %s} \\end{center}\n”,
          invoice_title );

boolean file_exists_boolean(
    statement_caption_logo_filename );

if ( file_exists_boolean() )
{
    ptr += sprintf(
                ptr,
                “\\begin{center}\n”
                “\\includegraphics{%s}\n”
                “\\end{center}\n”,
                statement_caption_logo_filename );
}

ptr += sprintf( ptr,
“\\begin{center}{%s} \\end{center}\n”,
          invoice_date_string );

/* Driver */
void invoice_latex_output(
    INVOICE_LATEX *invoice_latex );

/* Process */
FILE *appaserver_output_file(
    invoice_latex->
       statement_link->
       tex_filename );

fprintf( appaserver_output_file(),
          “%s\n”,
          invoice_latex->document_header );

fprintf( appaserver_output_file(),
          “%s\n”,
          invoice_latex->title );

fprintf( appaserver_output_file(),
          “%s\n”,
          invoice_latex->
             invoice_latex_table->
             display );

fprintf( appaserver_output_file(),
          “%s\n”,
          invoice_latex->footer );

void fclose( appaserver_output_file() );

void latex_tex2pdf(
    invoice_latex->
       statement_link->
       tex_filename,
    invoice_latex->
       statement_link->
       appaserver_link_working_directory );

printf(
    “%s<br>%s\n”,
    invoice_latex->
       statement_link->
       pdf_anchor_html,
    invoice_latex->
       statement_link->
       tex_anchor_html );
```

INVOICE_LATEX_EDUCATION

INVOICE_LATEX_TABLE (1)

```
/* Usage */
INVOICE_LATEX_TABLE *invoice_latex_table_new(
    enum invoice_enum invoice_enum,
    char *invoice_title,
    ENTITY_SELF *entity_self,
    CUSTOMER *customer,
    LIST *invoice_line_item_list,
    char *amount_due_label,
    INVOICE_SUMMARY *invoice_summary );

/* Process */
INVOICE_LATEX_TABLE *invoice_latex_table_calloc(
    void );

INVOICE_LATEX_ENTITY *
    self_invoice_latex_entity =
        invoice_latex_entity_new(
            entity_self->entity );

INVOICE_LATEX_ENTITY *
    customer_invoice_latex_entity =
        invoice_latex_entity_new(
            customer->entity );

LIST *invoice_latex_table_column_list(
    invoice_enum,
    invoice_summary->
        invoice_line_item_description_boolean,
    invoice_summary->
        invoice_line_item_discount_boolean,
    invoice_summary->
        invoice_line_item_quantity_decimal_count );

LIST *invoice_latex_table_row_list(
    invoice_line_item_list,
    invoice_latex_table_column_list() );

LATEX_TABLE *latex_table =
    latex_table_new(
        invoice_title,
        invoice_latex_table_column_list(),
        invoice_latex_table_row_list() );

char *latex_table_head_display(
    latex_table->begin_longtable,
    latex_table->latex_column_header_format_line,
    latex_table->latex_table_caption_display,
    latex_table->latex_column_header_text_line,
    latex_table->footer_display );

INVOICE_LATEX_SUMMARY *
    invoice_latex_summary_new(
        amount_due_label,
        invoice_summary->
            invoice_line_item_description_boolean,
        invoice_summary->
            invoice_line_item_discount_boolean,
        invoice_summary->
            invoice_line_item_extended_total,
        invoice_summary->
            invoice_line_item_discount_total,
        invoice_summary->invoice_amount,
        customer->journal_payable_balance,
        invoice_summary->amount_due );

char *invoice_latex_table_display(
    latex_table->caption_display,
    self_invoice_latex_entity->display,
    customer_invoice_latex_entity->display,
    latex_table_head_display(),
    latex_table->latex_row_list /* freed */,
    latex_table->end_longtable,
    invoice_latex_summary_new()->display );

/* Attributes */
INVOICE_LATEX_ENTITY *self_invoice_latex_entity;
INVOICE_LATEX_ENTITY *customer_invoice_latex_entity;
LIST *column_list;
LIST *row_list;
LATEX_TABLE *latex_table;
char *latex_table_head_display;
INVOICE_LATEX_SUMMARY *invoice_latex_summary;
char *display;
```



INVOICE_LATEX_TABLE (2)

```
/* Usage */
LIST *invoice_latex_table_column_list(
    enum invoice_enum invoice_enum,
    boolean invoice_line_item_description_boolean,
    boolean invoice_line_item_discount_boolean,
    int invoice_line_item_quantity_decimal_count );

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

LATEX_COLUMN *latex_column_new(
    “item_key” /* heading_string */,
    latex_column_text /* enum latex_column_enum */,
    0 /* float_decimal_count */,
    (char *)0 /* paragraph_size */,
    1 /* first_column_boolean */ );

list_set( column_list, latex_column_new() );

if ( invoice_line_item_description_boolean )
{
    char *invoice_latex_table_description_size(
        boolean invoice_line_item_discount_boolean );

    LATEX_COLUMN *latex_column_new(
        “description” /* heading_string */,
        latex_column_text /* enum latex_column_enum */,
        0 /* float_decimal_count */,
        invoice_latex_table_description_size()
            /* paragraph_size */,
        0 /* not first_column_boolean */ );

    list_set( column_list, latex_column_new() );
}

LATEX_COLUMN *latex_column_new(
    “quantity” /* heading_string */,
    latex_column_float /* enum latex_column_enum */,
    invoice_line_item_quantity_decimal_count
        /* float_decimal_count */,
    (char *)0 /* paragraph_size */,
    0 /* not first_column_boolean */ );

list_set( column_list, latex_column_new() );

if ( invoice_enum != invoice_workorder )
{
    LATEX_COLUMN *latex_column_new(
        “retail_price” /* heading_string */,
        latex_column_float /* enum latex_column_enum */,
        2 /* float_decimal_count */,
        (char *)0 /* paragraph_size */,
        0 /* not first_column_boolean */ );

    list_set( column_list, latex_column_new() );

    if ( invoice_line_item_discount_boolean )
    {
        LATEX_COLUMN *latex_column_new(
            “discount_amount” /* heading_string */,
            latex_column_float,
            2 /* float_decimal_count */,
            (char *)0 /* paragraph_size */,
            0 /* not first_column_boolean */ );

        latex_column_new()->dollar_sign_boolean = 1;

        list_set( column_list, latex_column_new() );
    }

    LATEX_COLUMN *latex_column_new(
        “extended_price” /* heading_string */,
        latex_column_float /* enum latex_column_enum */,
        2 /* float_decimal_count */,
        (char *)0 /* paragraph_size */,
        0 /* not first_column_boolean */ );

    latex_column_new()->dollar_sign_boolean = 1;

    list_set( column_list, latex_column_new() );
}
```

INVOICE_LATEX_TABLE (3)

```
/* Usage */
LIST *invoice_latex_table_row_list(
    LIST *invoice_line_item_list,
    LIST *invoice_latex_table_column_list );

/* Process */
LIST *latex_row_list = list_new();
int first_row_boolean = 1;

for INVOICE_ITEM_ITEM *invoice_line_item in
    invoice_line_item_list
{
    LIST *invoice_latex_table_cell_list(
        invoice_line_item->item_key,
        invoice_line_item->description,
        invoice_line_item->quantity,
        invoice_line_item->retail_price,
        invoice_line_item->discount_amount,
        invoice_line_item->extended_price,
        invoice_latex_table_column_list,
        first_row_boolean );

    LATEX_ROW *latex_row_new(
        invoice_latex_table_cell_list() );

    list_set( latex_row_list, latex_row_new() );

    first_row_boolean = 0;
}

/* Usage */
LIST *invoice_latex_table_cell_list(
    char *item_key,
    char *description,
    double quantity,
    double retail_price,
    double discount_amount,
    double extended_price,
    LIST *invoice_latex_table_column_list,
    boolean first_row_boolean );

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

LATEX_COLUMN *latex_column_seek(
    invoice_latex_table_column_list,
    "item_key" );

LATEX_CELL *latex_cell_new(
    latex_column_seek(),
    first_row_boolean,
    item_key /* datum */,
    0 /* not large_boolean */,
    0 /* not bold_boolean */ );

list_set( cell_list, latex_cell_new() );

LATEX_COLUMN *latex_column_seek(
    invoice_latex_table_column_list,
    "description" );

if ( latex_column_seek() )
{
    LATEX_CELL *latex_cell_new(
        latex_column_seek(),
        first_row_boolean,
        description /* datum */,
        0 /* not large_boolean */,
        0 /* not bold_boolean */ );

    list_set( cell_list, latex_cell_new() );
}

LATEX_COLUMN *latex_column_seek(
    invoice_latex_table_column_list,
    "quantity" );

LATEX_CELL *latex_cell_float_new(
    latex_column_seek(),
    first_row_boolean,
    quantity /* value */ );

list_set( cell_list, latex_cell_float_new() );

LATEX_COLUMN *latex_column_seek(
    invoice_latex_table_column_list,
    "retail_price" );

LATEX_CELL *latex_cell_float_new(
    latex_column_seek(),
    first_row_boolean,
    retail_price /* value */ );

list_set( cell_list, latex_cell_float_new() );

LATEX_COLUMN *latex_column_seek(
    invoice_latex_table_column_list,
    "discount_amount" );

if ( latex_column_seek() )
{
    LATEX_CELL *latex_cell_float_new(
        latex_column_seek(),
        first_row_boolean,
        discount_amount /* value */ );

    list_set( cell_list, latex_cell_float_new() );
}

LATEX_COLUMN *latex_column_seek(
    invoice_latex_table_column_list,
    "extended_price" );

LATEX_CELL *latex_cell_float_new(
    latex_column_seek(),
    first_row_boolean,
    extended_price /* value */ );

list_set( cell_list, latex_cell_float_new() );
```

INVOICE_LATEX_TABLE (4)

```
/* Usage */
char *invoice_latex_table_display(
    char *latex_table_caption_display,
    char *self_invoice_latex_entity_display,
    char *customer_invoice_latex_entity_display,
    char *latex_table_head_display,
    LIST *latex_table_latex_row_list /* freed */,
    char *latex_table_end_longtable,
    char *invoice_latex_summary_display );

/* Process */
char *latex_row_list_display(
    latex_table_latex_row_list /* freed */ );

snprintf(
    display[],
    sizeof ( display ),
    “%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s”,
    latex_table_caption_display,
    self_invoice_latex_entity_display,
    customer_invoice_latex_entity_display,
    latex_table_head_display,
    latex_table_row_list_display(),
    latex_table_end_longtable,
    invoice_latex_summary_display );
```

INVOICE_LATEX_SUMMARY

```
/* Usage */
INVOICE_LATEX_SUMMARY *
    invoice_latex_summary_new(
        char *amount_due_label,
        boolean invoice_line_item_description_boolean,
        boolean invoice_line_item_discount_boolean,
        double invoice_line_item_extended_total,
        double invoice_line_item_discount_total,
        double invoice_summary_invoice_amount,
        double customer_payable_balance,
        double invoice_summary_amount_due );

/* Process */
INVOICE_LATEX_SUMMARY *
    invoice_latex_summary_calloc(
        void );

char *invoice_latex_summary_extended_display(
    boolean invoice_line_item_description_boolean,
    boolean invoice_line_item_discount_boolean,
    double invoice_line_item_extended_total );

char *invoice_latex_summary_discount_display(
    boolean invoice_line_item_description_boolean,
    boolean invoice_line_item_discount_boolean,
    double invoice_line_item_discount_total );

char *invoice_latex_summary_amount_display(
    boolean invoice_line_item_description_boolean,
    boolean invoice_line_item_discount_boolean,
    double invoice_summary_invoice_amount );

char *invoice_latex_summary_payable_display(
    boolean invoice_line_item_description_boolean,
    boolean invoice_line_item_discount_boolean,
    double customer_payable_balance );

char *invoice_latex_summary_due_display(
    char *amount_due_label,
    boolean invoice_line_item_description_boolean,
    boolean invoice_line_item_discount_boolean,
    double invoice_summary_amount_due );

char *invoice_latex_summary_display(
    char *invoice_latex_summary_extended_display(),
    char *invoice_latex_summary_discount_display(),
    char *invoice_latex_summary_amount_display(),
    char *invoice_latex_summary_payable_display(),
    char *invoice_latex_summary_due_display() );

/* Usage */
char *invoice_latex_summary_empty_display(
    boolean invoice_line_item_description_boolean,
    boolean invoice_line_item_discount_boolean );

/* Process */

/* Attributes */
char *extended_display;
char *discount_display;
char *amount_display;
char *payable_display;
char *due_display;
char *display;
```

INVOICE_LATEX_ENTITY

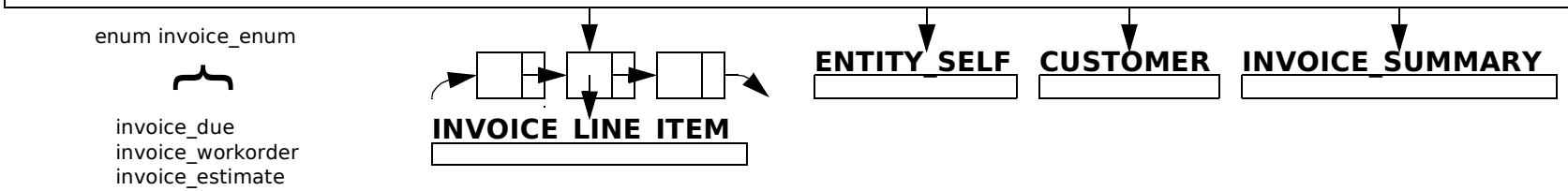
```
/* Usage */
INVOICE_LATEX_ENTITY *invoice_latex_entity_new(
    ENTITY *entity );

/* Process */
INVOICE_LATEX_ENTITY *invoice_latex_entity_calloc(
    void );

char *invoice_latex_entity_heading(
    entity->full_name );

char *escape_street_address =
    char *string_escape_array(
        INVOICE_LATEX_ESCAPE_ARRAY,
        entity->street_address );

if ( entity->city )
{
    char *invoice_latex_entity_city_state_zip(
        entity->city,
        entity->state_code,
        entity->zip_code );
}

char *invoice_latex_entity_display(
    entity->full_name,
    invoice_latex_entity_heading(),
    escape_street_address,
    invoice_latex_entity_city_state_zip() );

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

/* Process */
snprintf(
    heading[],
    sizeof ( heading ),
    "\\begin{tablular}[t]{l}\n"
    "\\bf %s:\n"
    "\\end{tabular}\n",
    full_name );

/* Usage */
char *invoice_latex_entity_city_state_zip(
    char *city,
    char *state_code,
    char *zip_code );

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

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

if ( state_code || zip_code )
{
    ptr += sprintf( ptr, "," );
}

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

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

/* Usage */
char *invoice_latex_entity_display(
    char *entity_full_name,
    char *invoice_latex_entity_heading,
    char *escape_street_address,
    char *invoice_latex_entity_city_state_zip() );

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

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

ptr += sprintf( ptr,
    "\\begin{tabular}{p{0.5in}l}\n" );

ptr += sprintf( ptr,
    "& %s \\\\\n"
    "& %s \\\\\n",
    entity_full_name,
    escape_street_address );

if ( latex_entity_city_state_zip )
{
    ptr += sprintf( ptr,
    "& %s\n",
    latex_entity_city_state_zip );
}

sprintf( ptr,
    "\\end{tabular}\n" );

/* Attributes */
char *heading;
char *escape_street_address;
char *city_state_zip;
char *display;
```

INVOICE (1)

```
/* Usage */
INVOICE *invoice_new(
    char *invoice_key,
    char *transaction_date_time_string,
    char *invoice_date_time_string,
    char *customer_full_name,
    char *customer_street_address,
    enum invoice_enum invoice_enum,
    LIST *invoice_line_item_list() );

/* Process */
INVOICE *invoice_calloc(
    void );

ENTITY_SELF *entity_self =
    entity_self_fetch(
        1 /* fetch_entity_boolean */ );

if ( !entity_self ) exit( 1 );
```

```
CUSTOMER *customer =
    customer_fetch(
        customer_full_name,
        customer_street_address,
        1 /* fetch_entity_boolean */,
        1 /* fetch_payable_balance_boolean */ );

if ( !customer ) exit( 1 );

/* Returns first non-empty parameter */
char *invoice_use_key(
    char *invoice_key,
    char *transaction_date_time_string,
    char *invoice_date_time_string );

char *invoice_title(
    char *customer_full_name,
    enum invoice_enum invoice_enum,
    char *invoice_use_key() );
```

```
char *invoice_date_string(
    char *invoice_date_time_string );

char *invoice_amount_due_label(
    enum invoice_enum invoice_enum );

INVOICE_SUMMARY *invoice_summary_new(
    LIST *invoice_line_item_list,
    customer->payable_balance );

/* Attributes */
LIST *invoice_line_item_list;
enum invoice_enum invoice_enum;
ENTITY_SELF *entity_self;
CUSTOMER *customer;
char *use_key;
char *title;
char *date_string;
char *amount_due_label;
INVOICE_SUMMARY *Invoice_summary;
```



INVOICE (2)

```
/* Driver */
void invoice_html_output(
    INVOICE *invoice,
    FILE *output_stream );

/* Process */
void invoice_html_output_style(
    FILE *output_stream );

void invoice_html_output_table_open(
    char *invoice->caption,
    FILE *output_stream );

void invoice_html_output_self(
    char *invoice->date_string,
    ENTITY_SELF *invoice->entity_self,
    boolean invoice->
       invoice_summary->
       invoice_line_item_description_boolean,
    boolean invoice->
       invoice_summary->
       invoice_line_item_discount_boolean,
    FILE *output_stream );

void invoice_html_output_customer(
    CUSTOMER *invoice->customer,
    boolean invoice->
       invoice_summary->
       invoice_line_item_description_boolean,
    boolean invoice->
       invoice_summary->
       invoice_line_item_discount_boolean,
    FILE *output_stream );

void invoice_html_output_table_header(
    enum invoice_enum invoice->invoice_enum,
    boolean invoice->
       invoice_summary->
       invoice_line_item_description_boolean,
    boolean invoice->
       invoice_summary->
       invoice_line_item_discount_boolean,
    FILE *output_stream );

void invoice_html_output_line_item_list(
    enum invoice_enum invoice->invoice_enum,
    LIST *invoice->invoice_line_item_list,
    boolean invoice->
       invoice_summary->
       invoice_line_item_description_boolean,
    boolean invoice->
       invoice_summary->
       invoice_line_item_discount_boolean,
    int invoice->
       invoice_summary->
       invoice_line_item_quantity_decimal_count,
    FILE *output_stream );

void invoice_html_output_footer(
    boolean invoice->
       invoice_summary->
       invoice_line_item_description_boolean,
    boolean invoice->
       invoice_summary->
       invoice_line_item_discount_boolean,
    double invoice->
       invoice_summary->
       invoice_line_item_extended_total,
    double invoice->
       customer->
       journal_payable_balance
          /* customer_payable_balance */,
    double invoice->invoice_summary->amount_due,
    char *invoice->amount_due_label,
    FILE *output_stream );

void invoice_html_output_table_close(
    FILE *output_stream );
```

INVOICE_SUMMARY

```
/* Usage */
INVOICE_SUMMARY *invoice_summary_new(
    LIST *invoice_line_item_list,
    double customer_payable_balance );

/* Process */
INVOICE_SUMMARY *invoice_summary_calloc(
    void );

boolean invoice_line_item_description_boolean(
    invoice_line_item_list );

boolean invoice_line_item_discount_boolean(
    invoice_line_item_list );

int invoice_line_item_quantity_decimal_count(
    invoice_line_item_list );

double invoice_line_item_extended_total(
    invoice_line_item_list );

double invoice_line_item_discount_total(
    invoice_line_item_list );

double invoice_summary_invoice_amount(
    double invoice_line_item_extended_total(),
    double invoice_line_item_discount_total() );

double invoice_summary_amount_due(
    double customer_payable_balance,
    double invoice_summary_invoice_amount() );

/* Attributes */
boolean invoice_line_item_description_boolean;
boolean invoice_line_item_discount_boolean;
int invoice_line_item_quantity_decimal_count;
double invoice_line_item_extended_total;
double invoice_line_item_discount_total;
double invoice_amount;
double amount_due;
```

INVOICE_LINE_ITEM

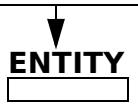
```
/* Usage */
LIST *invoice_line_item_list(
    char *customer_full_name,
    char *customer_street_address,
    char *sale_date_time );

/* Process */
char *invoice_line_item_system_string(
    customer_full_name,
    customer_street_address,
    sale_date_time );

FILE *appaserver_input_pipe(
    invoice_line_item_system_string() );

LIST *list = list_new();

while ( string_input( appaserver_input_pipe() ) )
{
    INVOICE_LINE_ITEM *invoice_line_item =
       invoice_line_item_parse(
          string_input() );

    list_set( list, invoice_line_item );
}
pclose( appaserver_input_pipe() );

/* Usage */
INVOICE_LINE_ITEM *invoice_line_item_parse(
    char *string_input() );

/* Process */
INVOICE_LINE_ITEM *invoice_line_item_new(
    strdup( item_key[] ),
    description[] /* stack memory */,
    atof( quantity[] ),
    atof( retail_price[] ),
    atof( discount_amount[] ) );

/* Usage */
INVOICE_LINE_ITEM *invoice_line_item_new(
    char *item_key,
    char *description /* stack memory */,
    double quantity,
    double retail_price,
    double discount_amount );

/* Process */
INVOICE_LINE_ITEM *invoice_line_item_calloc(
    void );

double invoice_line_item_extended_price(
    double quantity,
    double retail_price,
    double discount_amount );

/* Usage */
double invoice_line_item_extended_total(
    LIST *invoice_line_item_list );

/* Process */

/* Usage */
boolean invoice_line_item_description_boolean(
    LIST *invoice_line_item_list );

/* Process */

/* Usage */
boolean invoice_line_item_discount_boolean(
    LIST *invoice_line_item_list );

/* Process */

/* Usage */
int invoice_line_item_quantity_decimal_count(
    LIST *invoice_line_item_list );

/* Process */
for INVOICE_LINE_ITEM *invoice_line_item in
    invoice_line_item_list
{
    boolean float_is_integer(
       invoice_line_item->quantity );

    if ( !float_integer_boolean() )
    {
       return
       INVOICE_LINE_ITEM_QUANTITY_DECIMAL_COUNT;
    }
}
return 0;

/* Usage */
double invoice_line_item_discount_total(
    LIST *invoice_line_item_list );

/* Process */
for INVOICE_LINE_ITEM *invoice_line_item in
    invoice_line_item_list
{
    discount_total +=
       invoice_line_item->
          discount_amount;
}

/* Usage */
char *invoice_line_item_system_string(
    char *customer_full_name,
    char *customer_street_address,
    char *sale_date_time );

/* Process */
snprintf(
    system_string[],
    sizeof ( system_string ),
    "invoice_select.sh \"%s\" \"%s\" \"%s\"",
    customer_full_name,
    customer_street_address,
    sale_date_time );

/* Attributes */
char *item_key;
char *description;
double quantity;
double retail_price;
double discount_amount;
double extended_price;
```

CUSTOMER

```
/* Usage */
CUSTOMER *customer_fetch(
    char *customer_full_name,
    char *customer_street_address,
    boolean fetch_entity_boolean,
    boolean fetch_payable_balance_boolean );

/* Process */
char *entity_primary_where(
    char *customer_full_name,
    char *customer_street_address );

char *appaserver_system_string(
    char *CUSTOMER_SELECT,
    char *CUSTOMER_TABLE,
    char *entity_primary_where() );

char *string_pipe_input(
    appaserver_system_string() );

if ( !string_pipe_input() ) return NULL;

CUSTOMER *customer_parse(
    customer_full_name,
    customer_street_address,
    fetch_entity_boolean,
    fetch_payable_balance_boolean,
    string_pipe_input() );
```

```
/* Usage */
CUSTOMER *customer_parse(
    char *customer_full_name,
    char *customer_street_address,
    boolean fetch_entity_boolean,
    boolean fetch_payable_balance_boolean,
    char *string_pipe_input() );

/* Process */
CUSTOMER *customer_new(
    customer_full_name,
    customer_street_address );

if ( fetch_entity_boolean )
{
    ENTITY *entity =
        entity_fetch(
            customer_full_name,
            customer_street_address );

    if ( !entity ) exit( 1 );
}

if ( fetch_payable_balance_boolean )
{
    char *account_payable_string(
        ACCOUNT_PAYABLE_KEY,
        __FUNCTION__ );
```

```
    LIST *journal_entity_list(
        JOURNAL_SELECT,
        JOURNAL_TABLE,
        customer_full_name,
        customer_street_address,
        account_payable_string() );

    double journal_payable_balance =
        double journal_debit_credit_sum_difference(
            0 /* not element_accumulate_debit */,
            journal_entity_list() );
}

/* Usage */
CUSTOMER *customer_new(
    char *customer_full_name,
    char *customer_street_address );

/* Process */
CUSTOMER *customer_calloc(
    void );

/* Attributes */
char *customer_full_name;
char *customer_street_address;
boolean sales_tax_exempt_boolean;
ENTITY *entity;
double journal_payable_balance;
```



IMPORT_PREDICT (1)

```
/* Usage */
IMPORT_PREDICT *import_predict_new(
    const char *IMPORT_PREDICT_SQLGZ,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    char *appaserver_parameter->mount_point );

/* Process */
IMPORT_PREDICT *import_predict_calloc(
    void );

IMPORT_PREDICT_INPUT *import_predict_input_new(
    IMPORT_PREDICT_SQLGZ,
    application_name,
    mount_point );

char *import_predict_system_string(
    import_predict_input->filespecification );

char *import_predict_delete_role_system_string(
    const char *IMPORT_PREDICT_PROCESS_NAME,
    const char *ROLE_PROCESS_TABLE,
    const char *ROLE_SUPERVISOR );

char *import_predict_menu_href_string(
    const char *MENU_EXECUTABLE,
    char *application_name,
    char *session_key,
    char *login_name,
    char *role_name,
    boolean
      import_predict_input->
      application_menu_horizontal_boolean );

char *import_predict_menu_anchor_tag(
    const char *FRAMESET_PROMPT_FRAME,
    char *import_predict_menu_href_string() );

/* Usage */
char *import_predict_system_string(
    char *import_predict_input->filespecification );

/* Process */
snprintf(
    static system_string[],
    sizeof( system_string ),
    "zcat %s |"
    "mysqldump_sed.sh |"
    "sql.e 2>&1 |"
    "grep -vi duplicate",
    filespecification );

/* Attributes */
IMPORT_PREDICT_INPUT *import_predict_input;
char *system_string;
char *delete_role_system_string;
char *menu_href_string;
char *menu_anchor_tag;
```

IMPORT_PREDICT (2)

```
/* Driver */
if ( import_predict->
       import_predict_input->
       template_boolean )
{
   printf(
      "%s\n",
      IMPORT_PREDICT_TEMPLATE_MESSAGE );
}

if ( !import_predict->
       import_predict_input->
       entity_self )
{
   printf(
      "%s\n",
      IMPORT_PREDICT_ENTITY_SELF_MESSAGE );
   exit( 0 );
}

if (   import_predict->
         import_predict_input->
         exists_boolean
&&   !execute_boolean )
{
   printf(
      "%s\n",
      IMPORT_PREDICT_EXISTS_MESSAGE );
}

if ( execute_boolean )
{
   system( import_predict->system_string );

   system(
      import_predict->
         delete_role_system_string );

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

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

   printf(
      "%s\n",
         import_predict->menu_anchor_tag );

   printf( "<h3>Reminder:</h3>\n" );

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

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

   printf( "<h3>Process complete</h3>\n" );
}
else
/* Not execute_boolean */
{
   printf(
      "%s\n",
      IMPORT_PREDICT_SHORTCUT_MESSAGE );

   printf( "<h3>Hint message complete</h3>\n" );
}
```

IMPORT_PREDICT_INPUT

```
/* Usage */
IMPORT_PREDICT_INPUT *import_predict_input_new(
    const char *IMPORT_PREDICT_SQLGZ,
    char *application_name,
    char *mount_point );

/* Process */
IMPORT_PREDICT_INPUT *import_predict_input_calloc(
    void );

boolean import_predict_input_template_boolean(
    const char *APPLICATION_TEMPLATE_NAME,
    char *application_name );

boolean import_predict_input_exists_boolean(
    TRANSACTION_TABLE );
```

```
ENTITY_SELF *entity_self =
    entity_self_fetch(
        0 /* not fetch_entity_boolean */ );

char *import_predict_input_filespecification(
    const char *IMPORT_PREDICT_SQLGZ,
    char *mount_point );

boolean application_menu_horizontal_boolean(
    application_name );

/* Usage */
boolean import_predict_input_exists_boolean(
    const char *TRANSACTION_TABLE );
```

```
/* Process */
FOLDER *folder_where_fetch(
    TRANSACTION_TABLE /* folder_name */,
    (LIST *)0 /* role_attribute_exclude_lookup_..._list */,
    0 /* not fetch_folder_attribute_list */,
    0 /* not fetch_attribute */ );

(boolean)(unsigned int)(long)folder_where_fetch();

/* Attributes */
boolean template_boolean;
boolean exists_boolean;
ENTITY_SELF *entity_self;
char *filespecification;
boolean application_menu_horizontal_boolean;
```

ENTITY_SELF