Package Functions

Simply Smarter Information Management

Topic: Built-in Packages

We'll cover some useful included packages as well as talk about making your own packages.

UTIL/RPT:

- Control Functions
- Debugging Functions
- Get/Set CF Functions
- Get/Set Task Functions
- Locking CF Functions
- Trackor Functions
- Workplan Functions
- Utility Functions

DB Warmup

Rest Support

UTIL/RPT – Control Functions

OneVizion
Simply Smarter Information Management

SetFlag(FlagName, PayLoadData)
ClearFlag(FlagName)
IsFlagSet(FlagName)
GetFlagPayLoad(FlagName)

Notes:

- ***Flags*** are temporary session markers, like semaphores.
- This provides a method to skip or add functionality into a Rule based on other processes that may have happened prior to this Rule getting fired off.

Parameters:

- ***FlagName*** is a String representing a named semaphore
- ***PayloadData*** is an extra identifying piece of data if needed.

UTIL/RPT – Debug/Message Functions

RaiseError(msg)

SetMessage(Process, SubProcess, Message)

SendMail(Subject, Message, Sender, Recipient, cc, style, ReplyTo, bcc)

SendMailWithAttachments(Subject, Message, Sender, Recipient, cc, style, ReplyTo, AttachBlobDataId1, AttachBlobDataId2, AttachBlobDataId3, AttachBlobDataId4, AttachBlobDataId5, bcc)

GetAuditCallStack

Notes:

- ***RaiseError*** lets you raise an Oracle Error with the appropriate wrappings for the GUI to make a clean message to the User.
- ***SetMessage*** puts a record into the System Message table for debugging purposes.
- ***SendMail*** (***WithAttachments***) puts a record in the notifications table to be sent via email.
- ***GetAuditCallStack*** gets a string representation of the process stack in the current session. This can be used with any of the above to help debugging.

Parameters:

- ***Style*** defaults to "text", which gives a plain-text message. You can also choose "HTML", which will treat the ***Message*** text as HTML instead.

UTIL/RPT – Get/Set CF Functions

GetVal[DataType](KeyValue, FieldName)
GetVal[DataType]ByID(KeyValue, FieldID)

SetVal[DataType](KeyValue, FieldName, Val, IgnoreLocks)
SetVal[DataType]ByID(KeyValue, FieldID, Val, IgnoreLocks)

GetPrevVal(KeyValue, FieldName)
GetPrevValByID(KeyValue, FieldID)

GetPrevValLt(KeyValue, FieldName)
GetPrevValLtByID(KeyValue, FieldID)

Notes:

- ***[DataType]*** is the datatype format the user expects the function to return. These can be ***Str, Num, Date, Memo, Checkbox***. If the field is of a different type, the function will attempt to convert.
- ***GetPrevVal*** functions return the last value of this field for this trackor record, always as a String.
- ***GetPrevValLt*** functions do the same as ***GetPrevVal***, except they use AuditLog Lite tables, which are faster, but only have 30 days worth in them.

Parameters:

- ***KeyValue*** is the numeric system id for the record this Field's value is attached to. Can be TrackorID, WorkplanID, TaskID, WorkflowID or other.
- ***Val*** is the value to be set.
- ***IgnoreLocks*** lets the user force a change even if the field is locked for this record.

UTIL/RPT – Get/Set Task Functions

GetTaskDateBy[Index](WorkplanID, [Index], DateType, StartFinish, DateFormat)
GetTaskDateBy[Index]DateOnly(WorkplanID, [Index], DateType, StartFinish, NADate)

SetTaskDateBy[Index](WorkplanID, [Index], DateType, StartFinish, Val, ForceCalcs)
SetTaskDateBy[Index]DateOnly(WorkplanID, [Index], DateType, StartFinish, Val, ForceCalcs)

Notes:

- ***[Index]*** is the Task Level field used to identify which Task in the Workplan. Can be ***WBS*** or ***Ord*** (Order Number)
- Regular ***GetTaskDateBy*** functions return a string in the ***DateFormat*** specified, or "N/A" if the task is not applicable in this workplan. The ***DateOnly*** functions return a Date. If it is N/A, it returns the date given in ***NADate***.

Parameters:

- ***DateType*** specifies which Date Pair to use including Configured Date Pairs.
- ***StartFinish*** lets the user pick if Start o Finish date is required.
- ***ForceCalcs*** allows the user to either simply set the date, or set the date and cause cascading task calculations.
- ***Val*** is the value the date is set to.

UTIL/RPT – Other Task Functions

OneVizion
Simply Smarter Information Management

BlockTaskBy[Index](WorkplanID, [Index])
UnBlockTaskBy[Index] (WorkplanID, [Index])

SetTaskNABy[Index](WorkplanID, [Index])
UnSetTaskNABy[Index] (WorkplanID, [Index])

GetTask[Field](By[Index](WorkplanID, [Index])

Notes:

- ***[Index]*** is the Task Level field used to identify which Task in the Workplan. Can be ***WBS*** or ***Ord*** (Order Number)
- ***[Field]*** is the datapoint on the Task being requested. Can be ***Discp, Duration, BaselineDuration, Preds, Succs, or Comments***

- ***BlockTask*** functions set or unset the BlockCalcs setting.
- ***SetTaskNA*** functions set or unset the Tasks N/A

UTIL/RPT – Locking Functions

OneVizion
Simply Smarter Information Management

LockField(KeyValue, FieldName)
LockFieldByID(KeyValue, FieldID)
LockRelation(KeyValue, ParentTrackorType)
LockRelationByID(KeyValue, ParentTrackorTypeID)

UnLockField(KeyValue, FieldName)
UnLockFieldByID(KeyValue, FieldID)
UnLockRelation(KeyValue, ParentTrackorType)
UnLockRelationByID(KeyValue, ParentTrackorTypeID)

IsLocked(KeyValue, FieldName)
IsLockedByID(KeyValue, FieldID)
IsRelationLocked(KeyValue, ParentTrackorType)
IsRelationLockedByID(KeyValue, ParentTrackorTypeID)

Notes:

- Configured Fields can be locked or unlocked, which prevents users from changing the values in the fields unless they have special unlock privileges, and unlock it first.
- Relations between Trackors can also be locked and unlocked in this way.

Parameters:

- ***KeyValue*** is the numeric system id for the record this Field's value is attached to. Can be TrackorID, WorkplanID, TaskID, WorkflowID or other.

UTIL/RPT – Trackor and Workplan Functions

NewTrackor(TrackorType, TrackorKey, TrackorClass)
DropTrackor(KeyValue)
NewRelation(ParentID, ChildID)
AutoKeyGen(KeyValue)

NewWorkplan(WPTemplateName, KeyValue, WPName)
SetWPStart(WorkPlanID, StartDate)
SetWPFinish(WorkPlanID, FinishDate)

Notes:

- Some of these functions have additional parameter overloads to allow creation with slightly different parameter types. For instance, ***NewRelation()*** can also be called by naming Trackor Types and Trackor Keys.

Parameters:

- ***KeyValue*** is the numeric system id for the record this Field's value is attached to. Can be TrackorID, WorkplanID, TaskID, WorkflowID or other.

UTIL/RPT – Utility Functions

GetParentID(KeyValue, TrackorType)
GetTrackorID(XitorKey, TrackorType)

GetUN(UserID)
GetUserID(UserName)
GetUserTrackorID(UserName/UserID)
GetCurrentUser/ID/TrackorID

CopyEFile(BlobDataID, TargetKeyValue, TargetFieldID)
CopyEFileFromBlobData(BlobDataID, TargetKeyValue, TargetFieldID)

HasRole(SecRole, UserID)

IsNumber(str)

Simply Smarter Information Management

Notes:

- ***GetParentID*** lets the user grab the TrackorID of a direct ancestor to the given TrackorID.
- ***GetCurrentUser*** has several variants that let the user get userID, UserName, or the User's TrackorID.
- ***CopyEFile*** functions can copy from Blob_data table, or queue a copy from S3 via a java service if the File is already moved to S3.
- ***HasRole*** returns 1 if the given user has this Security Role.

Parameters:

- ***KeyValue*** is the numeric system id for the record this Field's value is attached to. Can be TrackorID, WorkplanID, TaskID, WorkflowID or other.

DB Warm-up

```
schedule_exec_warmup(
          p_db_warmup_run_name,
          p_exec_count,
          p_ttid,                   --Filter for TrackorType
          p_uid,                    --Filter for User
          p_start_date,             --Starting DateTime
          p_end_date,               --Ending DateTime
          p_page_name,              --Filter for Page
          p_min_db_runtime,         --Filter for DB Runtime
          p_dblink_name)
```

Notes:

- PKG_DB_WARMUP will run a selection of SQLs from the UsageLog.
- Schedules a DB Job to run this set of SQLs.
- Optionally can run across a DB Link to UsageLog in another system.

Parameters:

- p_db_warmup_run_name is a human readable name for this run.
- p_exec_count is number of repetition for the job
- p_dblink_name is the optional name of DB Link to use for the UsageLog

Rest Support

```
sql_to_xml(p_sql)
sql_to_json(p_sql [, p_is_array])

default_callback(
            p_http_status,
            p_response,
            p_tid,
            p_http_call_log_id,
            p_id2)
```

Notes:

- ***sql_to_xml*** runs a select query and returns a simple XML text matching the result columns and data.
- ***sql_to_json*** run the given select query and returns a simple json list [ ] with objects { } for each row. Optionally can return an empty string if no records are found.
- default_callback is the default handler for HTTP Call callbacks. It does nothing but throw an exception if the HTTP Status is not in the 200s.

Good Performing Code

- What are the best practices?

One key to make component migration easier is to write portable code when possible.

Some best practices to consider:

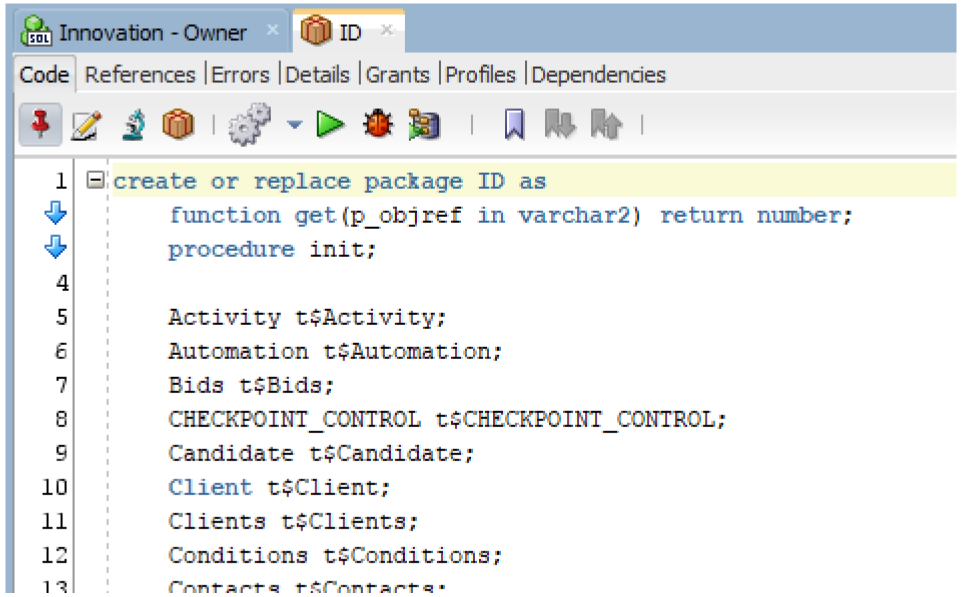
1. Comment your code heavily so that the next guy knows what you are intending to do. What is the purpose of this function? Where will it be used? Who asked for this new function? How are you accomplishing the goals in this function? These bits of documentation will give the next guy an idea of how to figure out any problems, or add new features.
2. Never use package functions in your where clauses.
3. Avoid **distinct**s and use **union all** over **union**s where possible.
4. IDs can greatly improve performance, but have the downside of not being portable between systems.
5. For PL/SQL blocks, consider selecting specific IDs into variables.
6. For SQL blocks, consider a with clause for IDs.

Object Reference (ID) Package

OneVizion
Simply Smarter Information Management

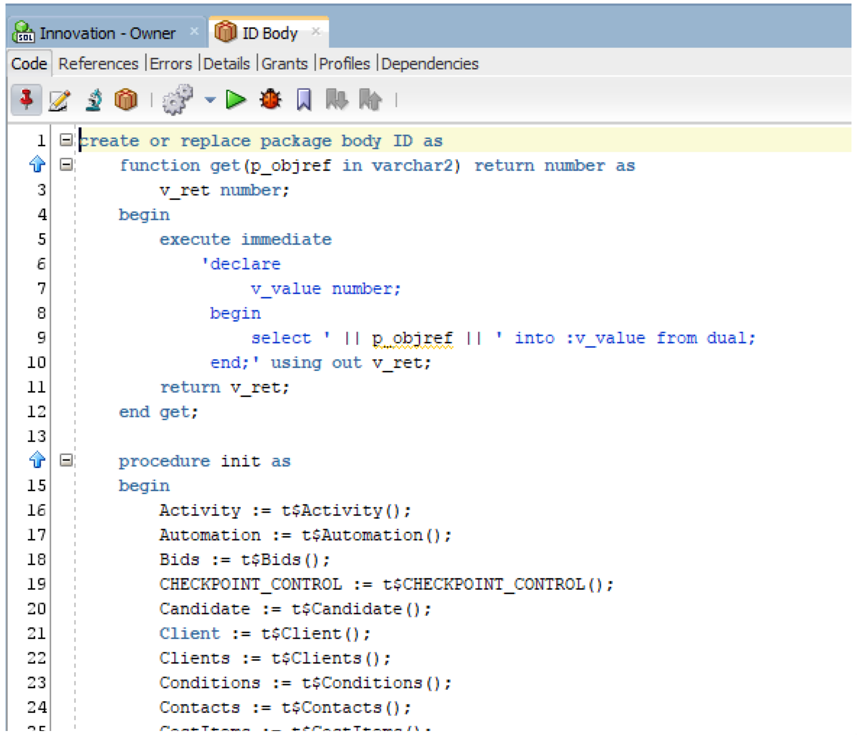
- What is it?

The ID package is an Oracle package that is built to allow you to get system-generated IDs for key objects like Trackor Types and Configured Fields using the admin-defined text.


Innovation - Owner
ID
Code | References | Errors | Details | Grants | Profiles | Dependencies

```
create or replace package ID as
    function get(p_objref in varchar2) return number;
    procedure init;

    Activity t$Activity;
    Automation t$Automation;
    Bids t$Bids;
    CHECKPOINT_CONTROL t$CHECKPOINT_CONTROL;
    Candidate t$Candidate;
    Client t$Client;
    Clients t$Clients;
    Conditions t$Conditions;
```


Innovation - Owner
ID Body
Code | References | Errors | Details | Grants | Profiles | Dependencies

```
create or replace package body ID as
    function get(p_objref in varchar2) return number as
        v_ret number;
    begin
        execute immediate
            'declare
                v_value number;
             begin
                select ' || p_objref || ' into :v_value from dual;
             end;' using out v_ret;
        return v_ret;
    end get;

    procedure init as
    begin
        Activity := t$Activity();
        Automation := t$Automation();
        Bids := t$Bids();
        CHECKPOINT_CONTROL := t$CHECKPOINT_CONTROL();
        Candidate := t$Candidate();
        Client := t$Client();
        Clients := t$Clients();
        Conditions := t$Conditions();
        Contacts := t$Contacts();
```

Object Reference (ID) Package

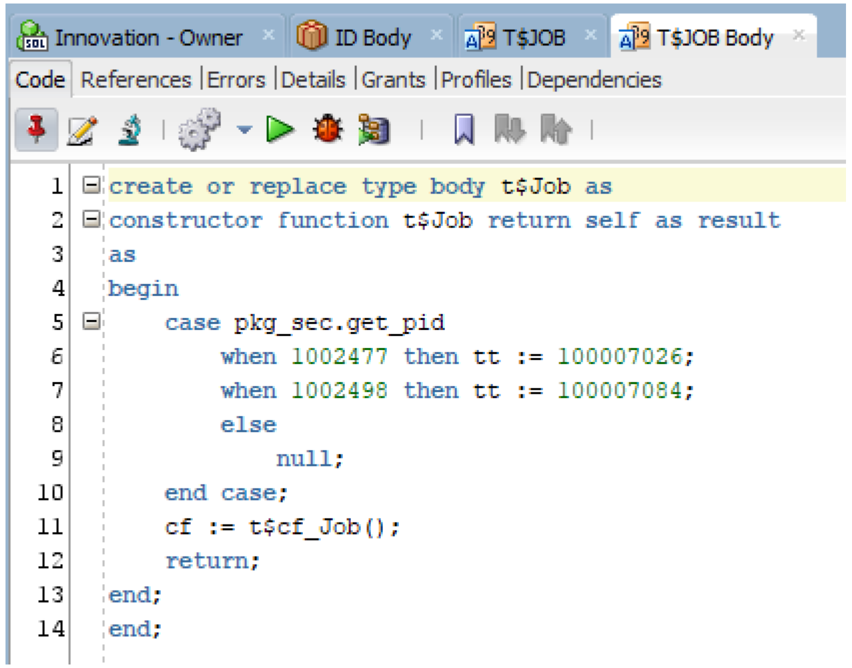
- How can we use it?

This is very easy to use!

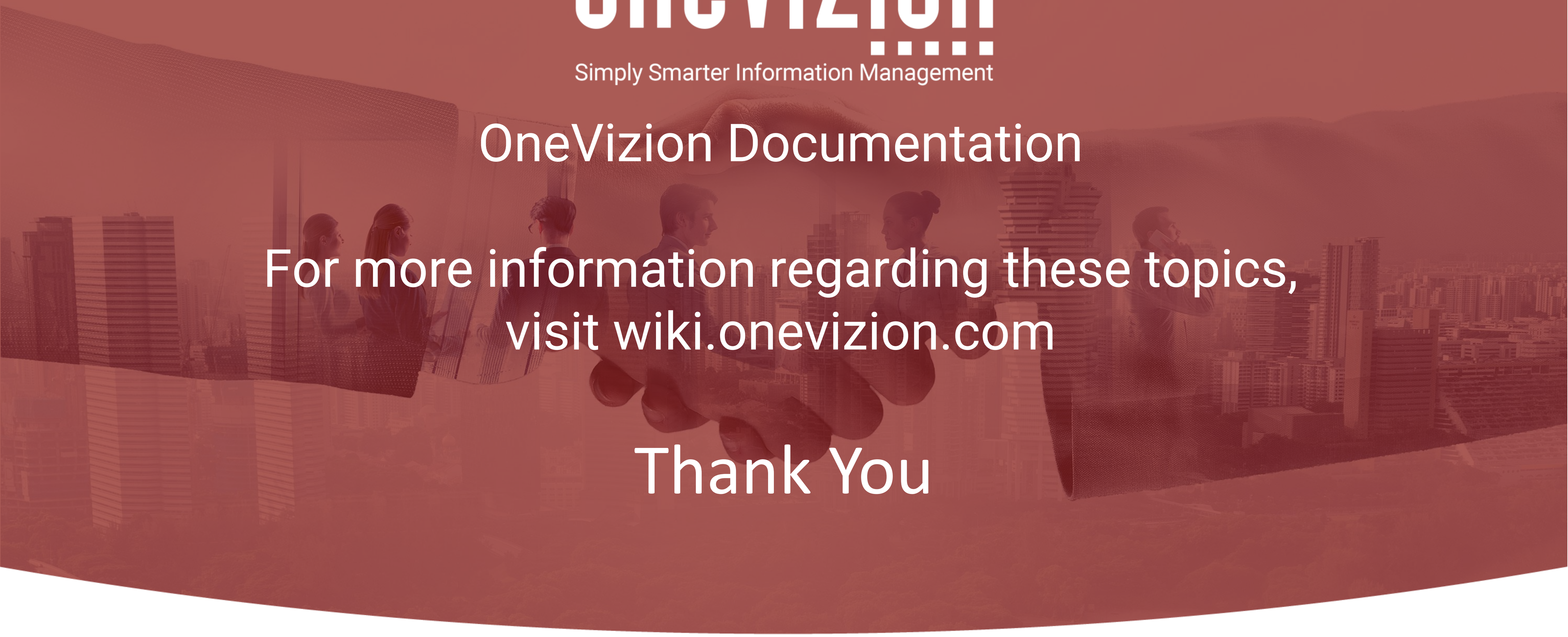
In PL/SQL (Rules, Imports, etc):

For the configured_field_id:
 id.<trackor_type>.cf.<config_field_name>
 id.job.cf.j_phase

For the xitor_type_id:
 id.<trackor_type>.tt
 id.job.tt



```
create or replace type body t$Job as
constructor function t$Job return self as result
as
begin
    case pkg_sec.get_pid
        when 1002477 then tt := 100007026;
        when 1002498 then tt := 100007084;
        else
            null;
    end case;
    cf := t$cf_Job();
    return;
end;
end;
```



OneVizion

Simply Smarter Information Management

OneVizion Documentation

For more information regarding these topics, visit wiki.onevizion.com

Thank You