For many eBS-DBA’s workflow is a strange and hardly understood module. Still it is widely used in 11i and 12i. So let’s dive into it’s workings in more detail.

This is part one of a series of 3. During this series, I used a 11.5.10 instance on a 9.2.0.8 database. The basics will still hold for earlier and later versions, but small modifications may be needed.

In this part we go into the definitions and the basics of Workflow. We start with some definitions, then build a simple basic workflow. And we see how this relates to the wf_tables in the database.

The basic terminology

First. What is a workflow? A workflow is a sequence of functions and events that follow a certain path based on decisions made during the progress of the sequence.

Most of us know the pictures from workflow builder. With the pictograms for functions joined together with lines.

That set is a definition of a workflow. That definition in Oracle is called a ‘process’. The nodes in the process can be functions, processes or notifications.

All these are grouped together in containers that Oracle called an ‘Itemtype’. The itemtype is very important, since it will become part of the primary key in the underlying tables.

The actual workflows that are running according to the definition of the itemtype are called ‘Item’s. The item is started as an itemtype and a process. And it is uniquely identified by itemtype and an itemkey.

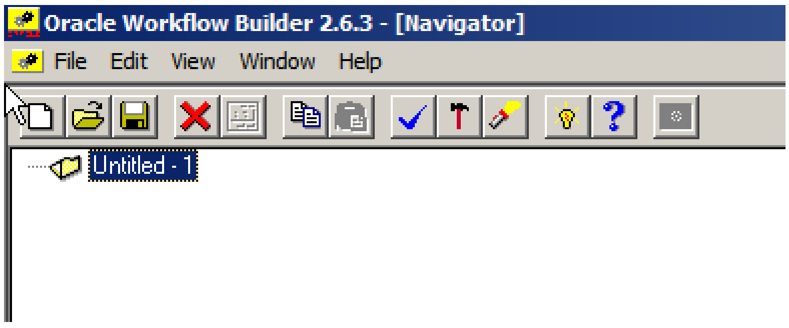
Every process consists of 2 or more nodes, that are joined together by transitions. At least 2 nodes are required, because a process needs a ’start’ and a ’stop’-node.

Ok. Enough talking. Let’s build a process and find out the rest along the way. By the way. All the definitions above will be linked to a glossary later on.

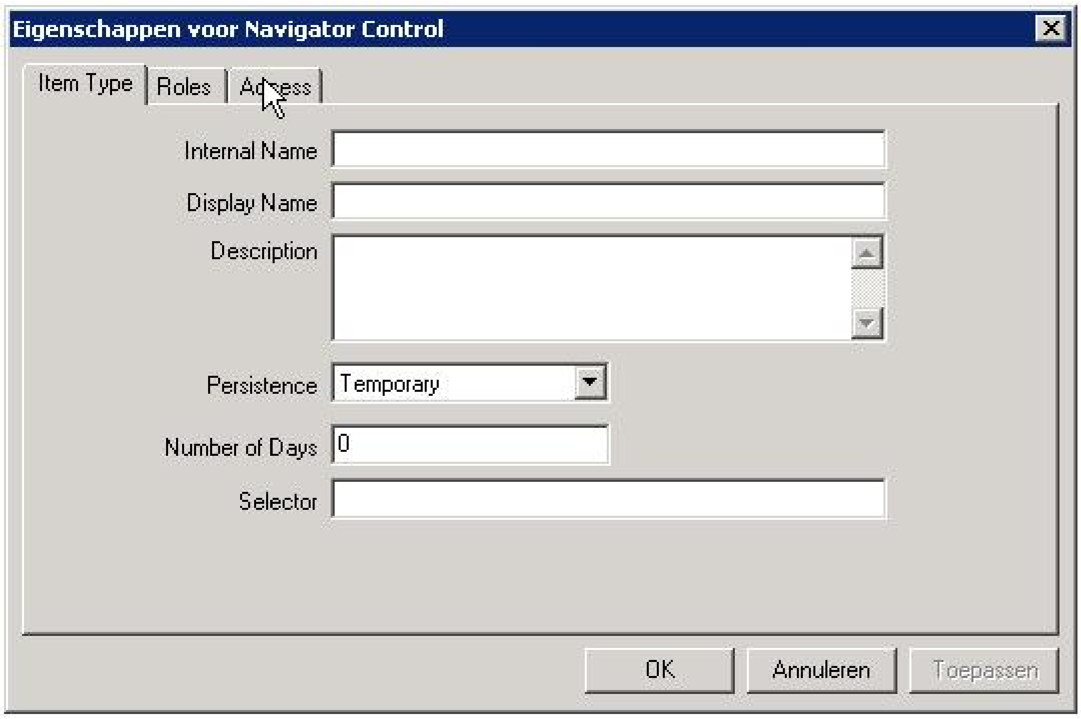
My first Itemtype

To start building our process, we first need the itemtype.

To create an itemtype, we use ‘Workflow builder’. In workflow builder, when we click the new button we are greeted with this screen:


Oracle Workflow Builder 2.6.3 - [Navigator]
File Edit View Window Help
Untitled - 1

On rightclicking the untitled map, we can create a new itemtype.



Internal Name	
Display Name	
Description	
Persistence	Temporary
Number of Days	0
Selector	

Internal name is the unique name that will be used in the table keys for this itemtype and it's items. It is limited to 8 characters. So choose wisely!

Display name is the name that will be shown to users when they need to interact with items from this itemtype.

The description…….. you can guess that one.

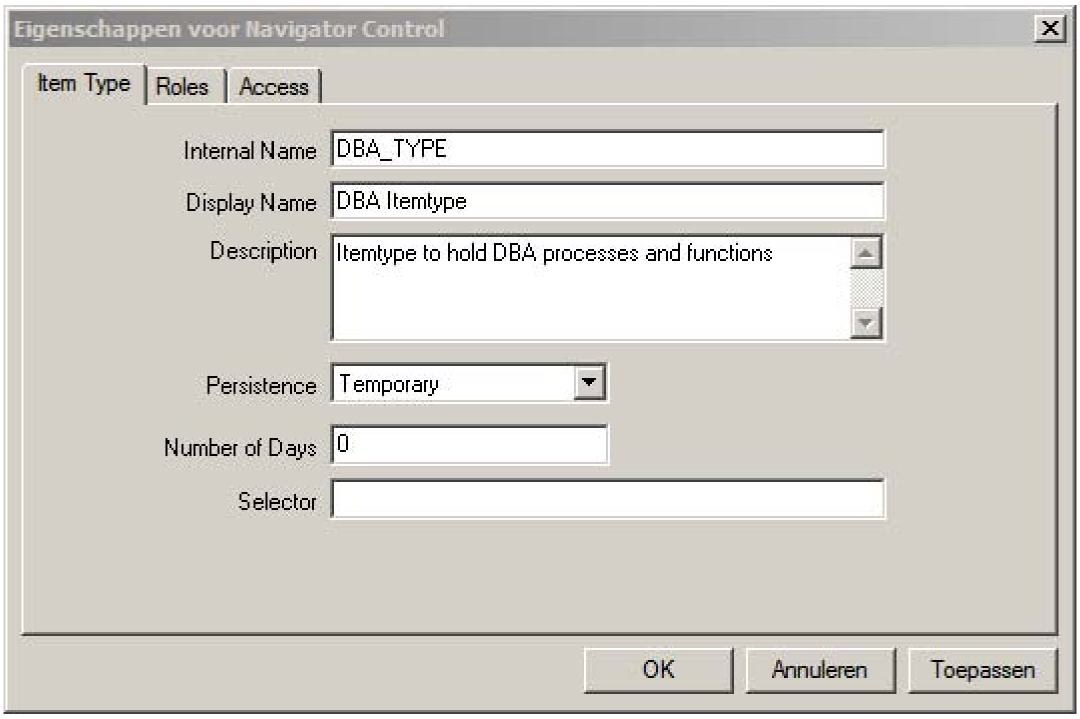
We will discuss the other three fields in a later article.

I choose to start building a flow that will do some DBA checks and tries to fix problems or notify the DBA if there is a problem. During the course of building this flow, we'll stumble on different features of the workflow engine.

The first step is to build the itemtype. I called it: DBA_TYPE.

With a display name: DBA Itemtype

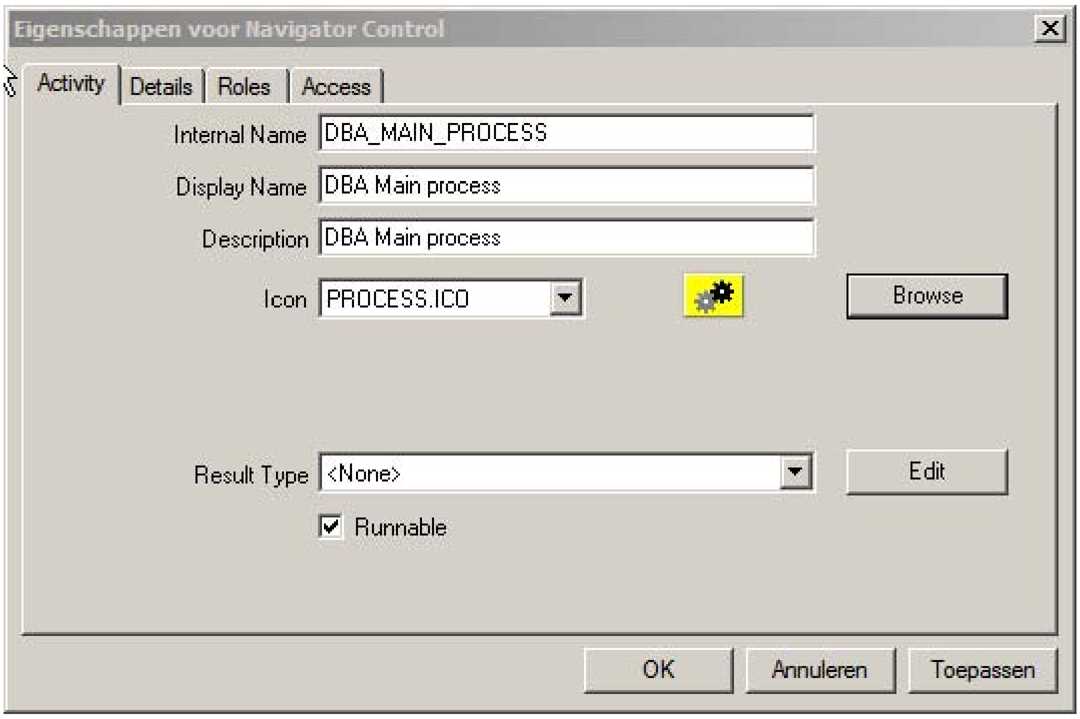
And a description: Itemtype to hold DBA processes and functions.



Internal Name: DBA_TYPE
Display Name: DBA Itemtype
Description: Itemtype to hold DBA processes and functions
Persistence: Temporary
Number of Days: 0
Selector:

When you open your newly created itemtype, you see the components that can be created within this itemtype.

You'll remember that the flow definition was called a processes. So next we create a new 'Process':



Internal Name: DBA_MAIN_PROCESS
Display Name: DBA Main process
Description: DBA Main process
Icon: PROCESS.ICO
Result Type: <None>
[✓] Runnable

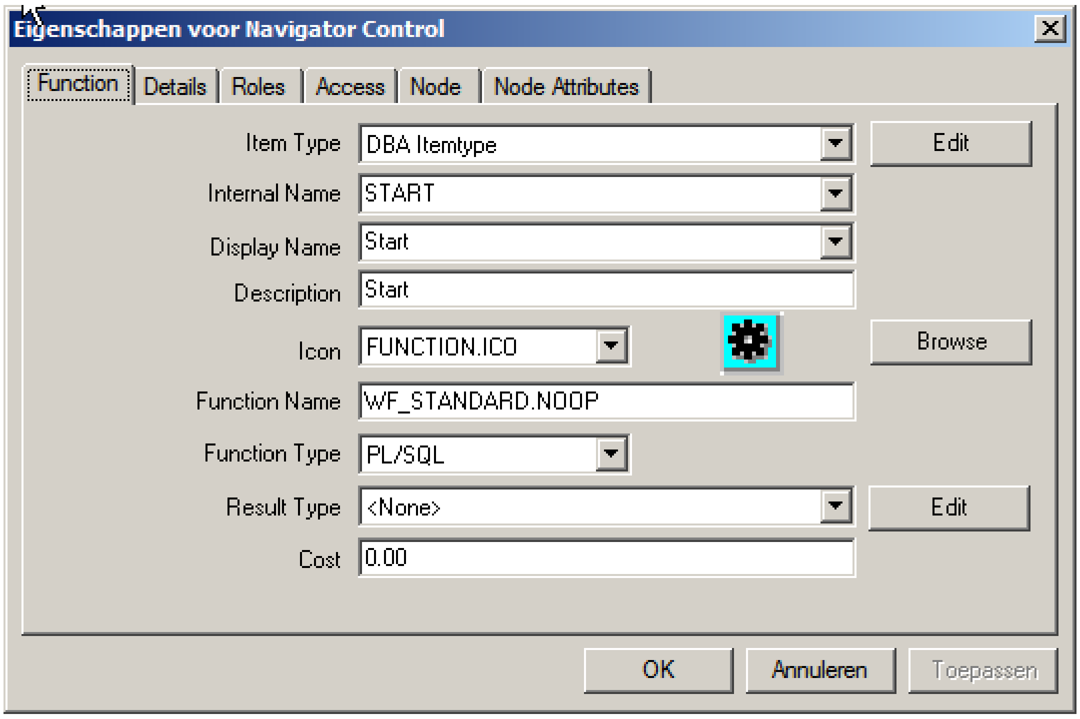
Because this is a process that we will only be calling from our client, we have no use for the result type at the moment. Later on, we'll see nested processes, where the result of a process will determine the direction of the calling process.

When we go to the Process Detail (right click the process). We again have a virgin drawing board. This will be where the actual flow is created.

Every process consists of activities (functions, notifications and processes) and the transitions between them (based on the results of the activities).

Also every process has to start with a 'Start' Activity and finish with an 'End' activity. (Take care to avoid loose ends, since the end finalizes the flow and gives back control, or makes the flow purgeable).

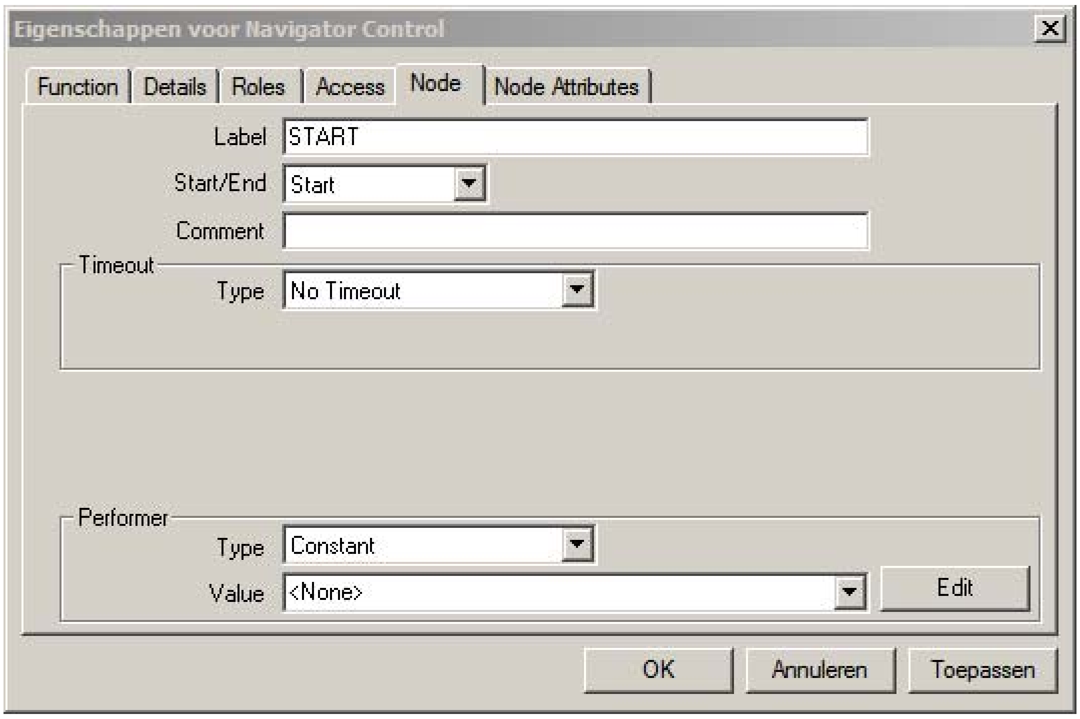
So first we create a new function to start our process.


Eigenschappen voor Navigator Control

Item Type	DBA Itemtype
Internal Name	START
Display Name	Start
Description	Start
Icon	FUNCTION.ICO
Function Name	WF_STANDARD.NOOP
Function Type	PL/SQL
Result Type	<None>
Cost	0.00

Note the wf_standard.noop for the function. This is a dummy function. Because the only purpose of this node is to indicate the starting point for the process.

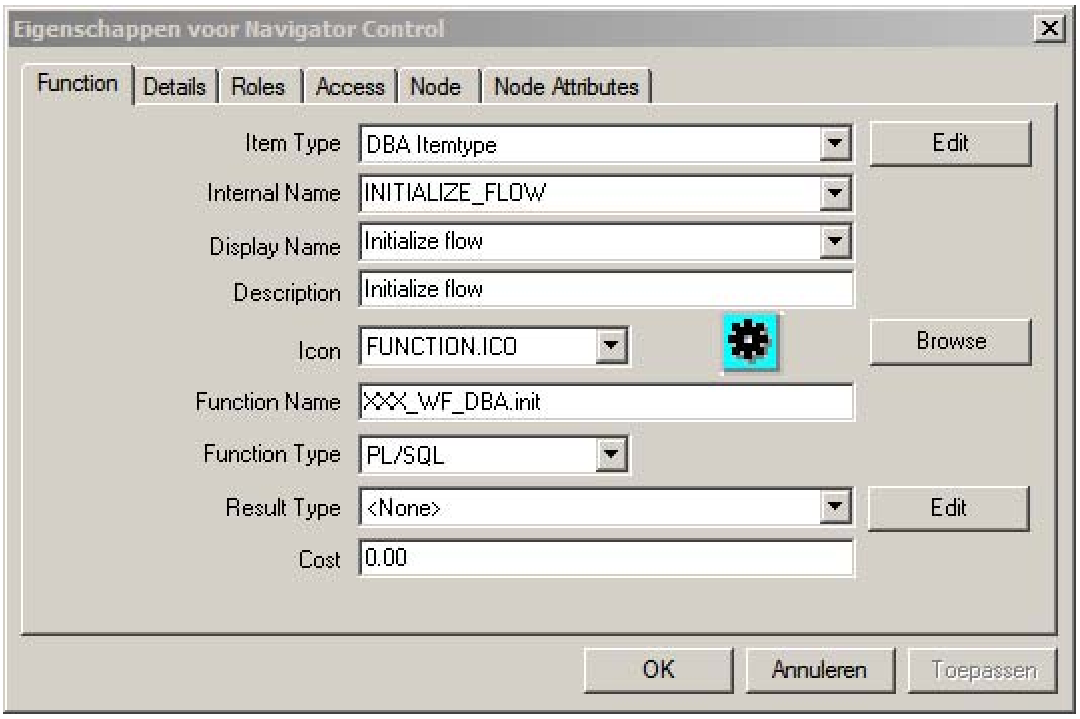
Even though we named this function 'START', we still need to flag it as a 'Start' function. That is in the node tab.


Eigenschappen voor Navigator Control

Label	START
Start/End	Start
Comment	
Timeout Type	No Timeout
Performer Type	Constant
Value	<None>

We then create an 'END' function in the same way.

Finally we create our own function.



Eigenschappen voor Navigator Control

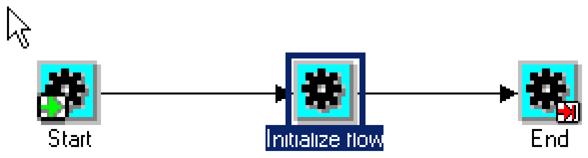
Function | Details | Roles | Access | Node | Node Attributes

Item Type	DBA Itemtype
Internal Name	INITIALIZE_FLOW
Display Name	Initialize flow
Description	Initialize flow
Icon	FUNCTION.ICO
Function Name	XXX_WF_DBA.init
Function Type	PL/SQL
Result Type	<None>
Cost	0.00

Now we have an item_type with 1 process, and 3 functions.

Time to connect the functions together. Right click START, and drag to INITIALIZE_FLOW. Then right click there and drag to END.

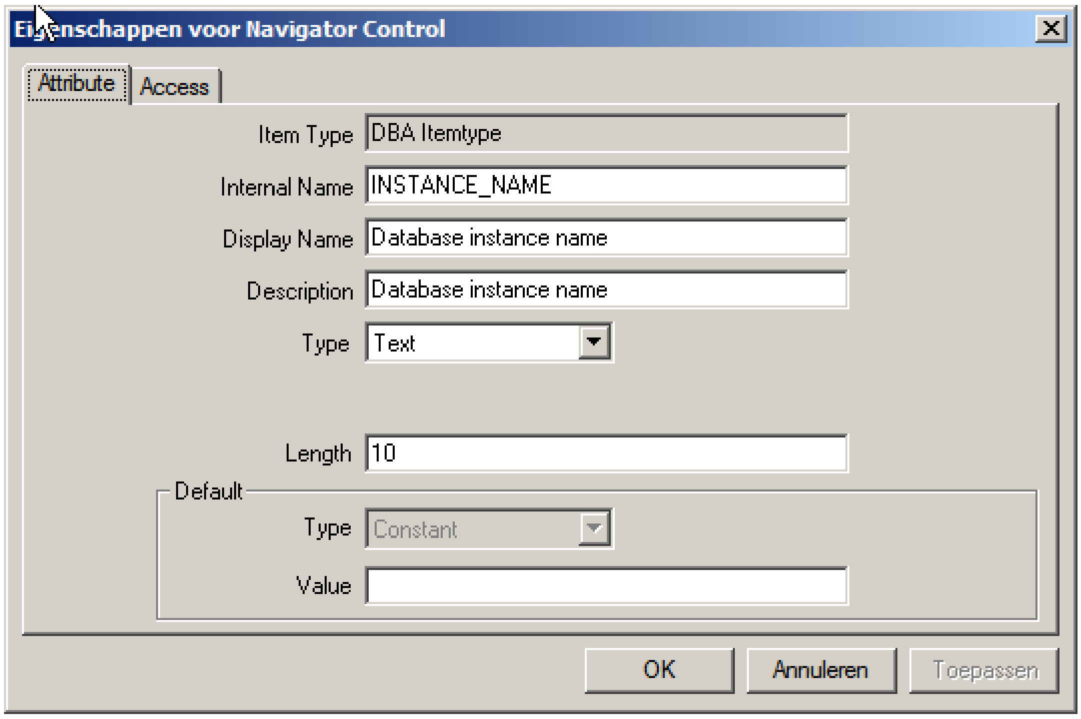
The result should be like:



Now we have a runnable flow. But there is more work to be done.

A process that only calls a function, is not really useful. Even with multiple functions, there need to be communication between them. So in come the 'Item Attributes'. Item Attributes are used as variables in the processes. As their name indicates, the values are bound to items. That means they are available all through a process, but the values are not available outside a particular item.

Let's create an item attribute to store our database instance name.



Item Type	DBA Itemtype
Internal Name	INSTANCE_NAME
Display Name	Database instance name
Description	Database instance name
Type	Text
Length	10

Default

Type	Constant
Value	

The item attribute has 2 interesting properties. The type. There is an extensive list of possible attribute types. Besides Text, Number and Date you can choose from Lookup, URL, Document, even Attribute and a few more. It will be a full article, to go through all the options.

Last thing remaining is to assign a value to this item attribute. This is done by our XXX_WF_DBA.init procedure.

The function to assign a value depends on the type of the item attribute. In this case we can use the function: WF_ENGINE.SetItemAttrText. For the other attribute types there are SetItemAttrNumber, SetItemAttrDate and SetItemAttrDocument. There are also Array versions of these, for bulk assignments.

We can now add this to our init function:

```
CREATE OR REPLACE PACKAGE BODY XXX_WF_DBA AS
  PROCEDURE init (p_item_type IN VARCHAR2
                 ,p_item_key IN VARCHAR2
                 ,p_actid IN NUMBER
                 ,p_funcmode IN VARCHAR2
                 ,p_result OUT VARCHAR2) IS
  Begin
       if p_funcmode='RUN' then
          wf_engine.SetItemAttrText(itemtype=>p_item_type
                                   ,itemkey =>p_item_key
                                   ,aname=>'INSTANCE_NAME'
                                 ,avalue=>sys_context('USERENV','DB_NAME'));
       end if;
       p_result:=wf_engine.eng_completed;
  END;
END XXX_WF_DBA;
```

```
/
```

The p_funcmode test will be explained later in the series. It is used to determine whether the function should execute, do nothing or rollback.

With this we created our own ‘hello world’ process. It doesn’t do anything useful, but it’s a working process.

It’s time to see how the different components are stored in the wf_* tables. And of course what happens when we create and execute an item. But that is part 2 of our series.

Continue reading here.