Implementing a Web Dynpro Callable Object which Implements the GP Interface
# Typographic Conventions

<table>
<thead>
<tr>
<th>Type Style</th>
<th>Represents</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example Text</strong></td>
<td>Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation.</td>
</tr>
<tr>
<td><strong>Example text</strong></td>
<td>Emphasized words or phrases in body text, graphic titles, and table titles.</td>
</tr>
<tr>
<td><strong>EXAMPLE TEXT</strong></td>
<td>Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text, for example, SELECT and INCLUDE.</td>
</tr>
<tr>
<td><strong>Example text</strong></td>
<td>Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.</td>
</tr>
<tr>
<td><strong>Example text</strong></td>
<td>Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.</td>
</tr>
<tr>
<td><strong>&lt;Example text&gt;</strong></td>
<td>Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.</td>
</tr>
</tbody>
</table>

## Icons

<table>
<thead>
<tr>
<th>Icon</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="caution_icon.png" alt="Caution Icon" /></td>
<td>Caution</td>
</tr>
<tr>
<td><img src="example_icon.png" alt="Example Icon" /></td>
<td>Example</td>
</tr>
<tr>
<td><img src="note_icon.png" alt="Note Icon" /></td>
<td>Note</td>
</tr>
<tr>
<td><img src="recommendation_icon.png" alt="Recommendation Icon" /></td>
<td>Recommendation</td>
</tr>
<tr>
<td><img src="syntax_icon.png" alt="Syntax Icon" /></td>
<td>Syntax</td>
</tr>
</tbody>
</table>
Contents

Business Scenario .................................................................................................................. 2
About This Document .......................................................................................................... 2
Applicable Releases ............................................................................................................. 3
Disclaimer ............................................................................................................................ 3
The Step By Step Solution .................................................................................................... 4
  Create local Web Dynpro DC and set dependencies ......................................................... 6
  Create the Web Dynpro Component .................................................................................. 14
  Create an instance of the callable object .......................................................................... 48
Business Scenario

In some cases it can be useful to implement a special functionality with a Web Dynpro Component which can be used within a callable object. For example if you need a highly sophisticated UI with some functionality within your application.

About This Document

This tutorial describes how to provide a Web Dynpro Component that can be used from a callable object in an action. The component needs to implement the Guided Procedures Interface.

You will implement a Web Dynpro Component for our time-off example, where you can enter 3 fields: start and end date and the absence type. You will also learn how to interact with input and output fields which can be used by the Guided Procedure runtime framework.

This tutorial is based on SAP Developing landscape settings. Please try to transfer some specific SAP landscape options to your own development landscape.

This document focuses on the API changes that have been conducted in the Guided Procedure API for SPS 7.

The changes can be summarized as follows:

- You need to reference to DC caf/eu/gp/api but instead of using the Public Part “default” you need to use Public Part “external”
- All published Interfaces now start with the prefix “IGP”
- All published Classes now start with the prefix “GP”

Prerequisites:

- CAF GP installed
- IDE (SAP NWDS) installed
- Connection and DTR user on NWDI productive system
- **Setup your development configuration:**
  Development Configuration that contains the Guided Procedures archives. Can be requested from your DTR track owner.

⚠️

This document explains how to implement a background callable object using SAP NetWeaver 2004s SPS07 API. It differs slightly from the APIs available with SPS06. If you need to migrate your callable objects from SAP NetWeaver SPS06, check SAP Note 924776.
This document was created at a point in time where the development track for Java Guided Procedure was homed in the development configuration Appl-645_dev and later within NW-Appl-700-SP_dev. At review time NW-Appl-700-VAL_cons was the current version which was used to proof this How-to. For Appl-645_dev or other development configurations in this document, please read the current version of the development configuration. The Development which holds Java-Guided Procedures is currently within the software component (SC) sap.com_SAP_EU_1.

The development done in this document will be placed in SC test.sap.com_SAP_EU-TST_1.

Be carefully when creating new development components (DCs); you cannot easily remove DCs from the development track!

The development component will be created as a local DC (enable the appropriate check mark during creation).

Applicable Releases
This tutorial is compatible with the following release" Beginning with SAP NetWeaver '04s SPS7".

Disclaimer
Any software coding and/or code lines / strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.
The Step By Step Solution
Before you begin developing, start your J2EE instance.

Set up the Development Configuration (Prerequisite)

1. Setup the Java Development Infrastructure:
   - Menu Window → Preferences
   - Enter the URL: http://sapnameserver:53600 under Java Development Infrastructure → Development Configuration Pool
   - With the button “Ping server” you may test the connection.

2. Open the perspectives “Development Configurations” and “Web Dynpro” using the menu Window → Open Perspective → Other…
   Two more buttons should appear on the upper right hand side.

3. Now change perspective to “Development Configurations” and open the “wizard for importing the development configuration” by pressing the button shown.
4. Click on “remote” and enter your DTR password.

5. Navigate to your track: (e.g. Netweaver → Applications → Appl-645_dev).

6. Click “Next”.

---

### Development Configuration Import

**Import of an existing development configuration**

- **Local development configuration file**: remote
- **Configuration**: User name: d03121
- **Remote configuration**: Password: 

### Development Configurations

<table>
<thead>
<tr>
<th>Development Configuration</th>
<th>Description</th>
<th>Internal Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appl-6452_dev</td>
<td></td>
<td>MAIN_Appl-6452_dev</td>
</tr>
<tr>
<td>Appl-6451_dev</td>
<td></td>
<td>MAIN_Appl-6451_dev</td>
</tr>
<tr>
<td>Appl-6450_dev</td>
<td></td>
<td>MAIN_Appl-6450_dev</td>
</tr>
<tr>
<td>Appl-6449_dev</td>
<td></td>
<td>MAIN_Appl-6449_dev</td>
</tr>
<tr>
<td>Appl-6448_dev</td>
<td></td>
<td>MAIN_Appl-6448_dev</td>
</tr>
</tbody>
</table>

---

**Actions:**
- Apply
- Next
- Finish
- Cancel
7. Click “Finish”.

8. Now the system will show entries under Active DCs, Local DCs, and Open Activities.

If you are using the SAP landscape, make sure that you never create activities on the current development core track, and always mark your development as local development.

Create local Web Dynpro DC and set dependencies
9. Change to perspective “Development Configurations”
   a. Switch to “Inactive DCs”
   b. Locate your component where you want to place your new DC (e.g. test.sap.com_SAP_EU-TST_1) and open the context menu with a right click
   c. Choose “Create New DC…”
10. Enter the following information
   - Vendor - testcustomer.com
   - Name - select a name from the drop down box if possible (caf, for example) and enter a unique name for your development component – eu/gp/ui/test/d031211, for example.
   - Caption – optionally, you can enter a caption
   - Language – e.g. American English
   - Domain – for example, Basis

   Enable the option Local Development Component if you want to create a test only. Please refer to the remark in the prerequisites section.

11. Select Type “Web Dynpro” and check “Sync Used Archives”
   a. Click Next
   b. Click Finish
12. Change to the ‘Web Dynpro’ Perspective and click on tab ‘Web Dynpro Explorer’
   
a. Now expand the tree of your previously created DC and navigate to DC MetaData → DC Definition → Used DCs
   
b. Right click on the node “Used DCs” and click on Add Used DC…
13. Add dependency to the Public Parts of DC "caf.eu/gp/api"
   a. In this example from tab ‘Active’ open the following tree:
      NW-Appl-700-SP_dev [0] →
      sap.com_SAP-EU_1 →
      caf.eu/gp/api → DC MetaData → Public Parts → external
   b. For “Dependency Type” select Build Time, Deploy Time, and Run Time.
   c. Click Finish.
   Make sure the dependency has been added successfully.
14. Next, check that:

- you can find the entry under “Used DCs” and
- the properties for: “At Build Time”, “At Deploy Time” and “At Run Time” are set to “true”.

To change the dependency type:

a. Open the context menu of the entry ca/eu/gp/api/barb2right “Edit Used DC”.

b. Make your changes and click Finish.

15. Now continue in the same way to add dependencies for “caf/eu/gp/api/wd” as follows:

a. Select Appl-645_dev [0] → sap.com_SAP-EU_1 → caf/eu/gp/api/wd → DC MetaData → Public Parts → GPWebDynproCO.

b. For “Dependency Type” select Build Time, Deploy Time, and Run Time.

c. Click Finish.
16. Define Web Dynpro References for your DC
   a. Right click on your DC
   b. Click Properties.

17. Select entry Web Dynpro References
   a. Select tab page Library References
   b. Choose Add
c. Enter the reference caf~eu~gp~api
18. Save your work regularly in the Web Dynpro perspective.

- You will recognize from the asterisk (*) in front of your DC that something has been changed.
- With the Button “Save all Metadata” you can save your changes.
Create the Web Dynpro Component

19. Open the node Web Dynpro:
   a. Right click Web Dynpro Components
      • Click Create Web Dynpro Component

20. Enter the basic data for the Web Dynpro Component:
    • Component Name, e.g. CCallableObject1
    • Component Package, e.g. com.testcustomer. caf.eu.gp.ui.test
    • Window Name, e.g. WCallableObject1
    • Window Package, e.g. com.testcustomer. caf.eu.gp.ui.test
    • View Name, e.g. VCallableObject1
    • View Package, e.g. com.testcustomer. caf.eu.gp.ui.test
    Click Finish.
21. In the ‘Web Dynpro Explorer’ expand the tree of your Web Dynpro Component.
   - Now expand the tree of your previously created DC and navigate to Web Dynpro Component → CCallableObject1 → Component Interface
   - Right click on the node Implemented Interfaces and click on Add.

22. Choose IGPWebDynproCO – com.sap.caf.eu.gp.co.webdynpro and click OK
    Note: If you cannot see this entry you have not added the used DC correctly.
23. Click Finish.

24. The following parts have been generated:
   - InterfaceViews ‘WebDynproCOInterfaceView’
   - Windows ‘WebDynproCO’
   - Within the interface controller two new methods have been generated:
     - execute()
     - getDescription()
25. With a double click on “Interface Controller” you will see the methods generated within the interface controller. Use the tab “Methods” and “Events”.

- **void execute**
  (com.sap.caf.eu.gp.co.api.IGPExecutionContext executionContext)

- **com.sap.caf.eu.gp.co.api.IGPTechnicalDescription getDescription**(java.util.Locale locale)

and the following event has been generated:

- **TechnicalException**
  (com.sap.caf.eu.gp.exception.api.GPTechnicalCallableObjectException technicalException)
26. Implementation of getDescription()

a. Double click on node Interface Controller or choose Edit from the context menu.

b. Change to tab Implementation

c. Position to method getDescription(Locale) from the Outline View.
27. Now implement the method. You may copy the code from the section below.

With Ctrl+Shift+O you can organize your imports (the imports are also shown below).
public com.sap.caf.eu.gp.co.api.IGPTechnicalDescription getDescription( java.util.Locale locale )
{
//@@begin getDescription()
IGPTechnicalDescription technicalDescription;
IWDTTextAccessor textAccessor = wdComponentAPI.getTextAccessor();
GPWebDynproResourceAccessor resourceAccessor = new GPWebDynproResourceAccessor(textAccessor);
technicalDescription = GPCallableObjectFactory.createTechnicalDescription(
   "CO_NAME_KEY",
   "CO_DESCRIPTION_KEY",
   resourceAccessor,
   locale);

// Define Input and Output Structure
IGPStructureInfo input = technicalDescription.getInputStructureInfo();
IGPStructureInfo output = technicalDescription.getOutputStructureInfo();

try {
// Define Input Parameters
IGPAttributeInfo inputStartDate = input.addAttribute(
   "I_START_DATE",
   IGPAttributeInfo.BASE_DATE);
inputStartDate.setMultiplicity(IGPAttributeInfo.MULTIPLICITY_1_1);

IGPAttributeInfo inputEndDate = input.addAttribute(
   "I_END_DATE",
   IGPAttributeInfo.BASE_DATE);
inputEndDate.setMultiplicity(IGPAttributeInfo.MULTIPLICITY_1_1);

IGPAttributeInfo inputAbsenceType = input.addAttribute(
   "I_ABSENCE_TYPE",
   IGPAttributeInfo.BASE_STRING);
inputAbsenceType.setMultiplicity(IGPAttributeInfo.MULTIPLICITY_0_1);

// Define Output Parameters
IGPAttributeInfo outputStartDate = output.addAttribute(
   "O_START_DATE",
   IGPAttributeInfo.BASE_DATE);
outputStartDate.setMultiplicity(IGPAttributeInfo.MULTIPLICITY_1_1);

IGPAttributeInfo outputEndDate = output.addAttribute(
   "O_END_DATE",
IGPAttributeInfo.BASE_DATE);
outputEndDate.setMultiplicity(IGPAttributeInfo.MULTIPLICITY_1_1);

IGPAttributeInfo outputAbsenceType = output.addAttribute("O_ABSENCE_TYPE",

IGPAttributeInfo.BASE_STRING);
outputAbsenceType.setMultiplicity(IGPAttributeInfo.MULTIPLICITY_0_1);

// Define Result State
IGPCOResultStateInfo resultStateSuccess
    = technicalDescription.addResultState("RESULT_NAME_SUCCESS");
resultStateSuccess.setNameKey("SUCCESS_KEY");
resultStateSuccess.setDescriptionKey("SUCCESS_DESCRIPTION_KEY");

// Define Exceptions
IGPExceptionInfo disaster
    = technicalDescription.addProcessException("EXCEPTION_NAME_DISASTER");

disaster.setNameKey("DISASTER_KEY");
disaster.setDescriptionKey("DISASTER_DESCRIPTION_KEY");
disaster.setFatal(true);

} catch (GPInvocationException e) {
    return null;
}

return technicalDescription;
//@@end

The following import statements have been added:

//@@begin imports
import java.sql.Date;
import com.sap.caf.eu.gp.co.api.GPCallableObjectFactory;
import com.sap.caf.eu.gp.co.api.GPStandardResourceAccessor;
import com.sap.caf.eu.gp.co.api.IGPCOResultStateInfo;
import com.sap.caf.eu.gp.co.api.IGPExceptionInfo;
import com.sap.caf.eu.gp.co.api.IGPTechnicalDescription;
import com.sap.caf.eu.gp.exception.api.GPInvocationException;
import com.sap.caf.eu.gp.structure.api.IGPAttributeInfo;
import com.sap.caf.eu.gp.structure.api.IGPStructure;
import com.sap.caf.eu.gp.structure.api.IGPStructureInfo;
import com.testcustomer.caf.eu.gp.ui.test.wdp.ICallableObject1Interface;
import com.testcustomer.caf.eu.gp.ui.test.wdp.ICallableObject1Interface.IContextElement;
//@@end
28. In the Message Pool add messages for keys used in the interface controller (method getDescription).
   - CO_NAME_KEY
   - CO_DESCRIPTION_KEY

29. Navigate to Web Dynpro Components → CallableObject1 → Message Pool
   a. Right click and choose “Open Message Editor” or double-click.
   b. A new window will appear where you can add messages.
   c. Click on Add and choose the message type “text” and message text type “not_specified”.
30. Now implement the method `execute()` in a same way as before

- You may copy the code from the section below
- With Ctrl+Shift+O you can organize your imports (the imports are also mentioned below)
- Be careful to use `import java.sql.Date;` for the items of type date
- Don’t forget to save your project from time to time.

**Note:**
If you copy the coding below, some errors will appear because the context still does not contain the `executionContext`. This will be fixed in the next steps by creating the context attributes and mapping them to the context of the interface controller. If this is done and you build the component the errors will disappear.
public void execute(com.sap.caf.eu.gp.co.api.IGPExecutionContext executionContext)
{
//@@begin execute()
IContextElement contextElement = wdContext.currentContextElement();
contextElement.setExecutionContext(executionContext);

// define Input Structure
IGPStructure input = executionContext.getInputStructure();
try
{
// output == input
Date start = new Date(input.getAttributeAsDate("I_START_DATE").getTime());
contextElement.setStartDate(start);

Date end = new Date(input.getAttributeAsDate("I_END_DATE").getTime());
contextElement.setEndDate(end);

String absenceTyp = input.getAttributeAsString("I_ABSENCE_TYPE");
contextElement.setAbsenceType(absenceTyp);
contextElement.setExecutionContext(executionContext);
}
catch (GPInvocationException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}

//@@end
}

The following import statements have been additionally added:

//@@begin imports
import java.sql.Date;
import com.sap.caf.eu.gp.co.api.GPCallableObjectFactory;
import com.sap.caf.eu.gp.co.api.GPStandardResourceAccessor;
import com.sap.caf.eu.gp.co.api.IGPCOResultStateInfo;
import com.sap.caf.eu.gp.co.api.IGPExceptionInfo;
import com.sap.caf.eu.gp.co.api.IGPTechnicalDescription;
import com.sap.caf.eu.gp.exception.api.GPInvocationException;
import com.sap.caf.eu.gp.structure.api.IGPAttributeInfo;
import com.sap.caf.eu.gp.structure.api.IGPStructure;
import com.sap.caf.eu.gp.structure.api.IGPStructureInfo;
import com.testcustomer.caf.eu.gp.ui.test.wdp.IPrivateCCallableObject1Interface;
import com.testcustomer.caf.eu.gp.ui.test.wdp.IPublicCCallableObject1Interface.IContextElement;
//@end
31. In the Component Controller add context attributes for Start and End Date, the Absence Type and the execution context.

a. Right click on “Component Controller” and choose **Edit** or double click on the node “Component Controller”

b. Change to the **Context** tab

c. Right click on Context

d. Choose **New**

e. Choose **Value Attribute**

f. Now enter an appropriate attribute name, e.g. startDate, endDate, absenceType and executionContext

g. In properties, edit the type of attribute. For the *date attributes change to date. For absenceType you may use string.
32. For the ExecutionContext you need a Java native type
   a. Click on the button.
   b. Choose **Java Native Type** and click on **Browse...**
   c. In the next window enter "IGPExe", the search list will be reduced to IGPExecutionContext
   d. Mark the entry and click **OK**
   e. Now com.sap.caf.eu.gp.co.api.IGPExecutionContext will be displayed in the field Java Native Type.
   f. Click **OK**.
33. Now you need to define the context mapping between the component controller and the interface controller.

   a. Right click on node CCallableObject1 and select **Open Data Modeler** or a double click on the same node will open the data modeler

   b. Click on “create a datalink” and then drag the mouse from the Interface Controller to the Component Controller.

The cursor will change its shape to a plug icon, while you are connecting the two controllers. Then a window will be displayed where you can create new maps between the interface and the component controller.
34. Now drag the elements (one by one) from the Interface Controller (right side) to the Component Controller (left side). Be careful to release your mouse button over the node otherwise it will not work.

   a. Drag and drop the attribute absenceType. A new window will appear
   b. Check mark the attribute and click OK
   c. Repeat the previous steps for all attributes
   d. Now click Finish.

35. In the diagram view you see a connecting line between interface and component controller, which indicates that a mapping has been carried out.

   In the “context” tab of your Web Dynpro Component CCallableObject1 you will see the newly created attributes and the mapping indicated with a small arrow over the icon \[\text{context}\] in front of the attribute.
When you save and build the DC all errors within the interface coding controller should disappear.

36. If you want to change the mappings, you double-click on the connecting line or on CalableObject1.
37. The next step is to create context attributes at the side of the view. The steps are the same as before.
   a. Drag and drop a “Data Link” from the View VCallableObject1 to the Component Controller and create the attributes with their mapping as before
   b. Now you will be able to access the values within the view and are able to create a simple view.

38. In the component controller add method ‘complete()’
   a. Double click the node Component Controller
   b. Change to tab Methods
   c. Add the method complete() with return type void (click New and check Method on the next screen)
   d. Click Finish.
39. Now implement method `complete()`:
   a. Change to tab Implementation and position with the outline view to the newly created method stub.
   b. Write output parameters in the execution context.
   c. Set defined result state in execution context.
   d. Complete processing of the callable object.
   e. With Ctrl+Shift+O you can organize your imports (the imports are also mentioned below).
public void complete()
{
    //@@begin complete()
    IContextElement contextElement = wdContext.currentContextElement();
    IGPExecutionContext ExecutionContext = contextElement.getExecutionContext();

    // define Output Structure
    IGPStructure output = ExecutionContext.getOutputStructure();

    try {
        // set output
        output.setAttributeValue("O_START_DATE", contextElement.getStartDate());
        output.setAttributeValue("O_END_DATE", contextElement.getEndDate());
        output.setAttributeValue("O_ABSENCE_TYPE", contextElement.getAbsenceType());

        // success
        ExecutionContext.setResultState("RESULT_NAME_SUCCESS");
        ExecutionContext.processingComplete();
    }
    catch (GPInvocationException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (GPEngineException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    //@@end
}

Import statements:

//@@begin imports
import com.sap.caf.eu.gp.co.api.IGPExecutionContext;
import com.sap.caf.eu.gp.exception.api.GPEngineException;
import com.sap.caf.eu.gp.exception.api.GPInvocationException;
import com.sap.caf.eu.gp.structure.api.IGPStructure;
import com.testcustomer.caf.eu.gp.ui.test.wdp.IPrivateCallableObject1;
import com.testcustomer.caf.eu.gp.ui.test.wdp.IPublicCallableObject1.IContextElement;
//@@end
40. Edit view ‘VCallableObject1’. Locate the node VCallableObject1 and Double Click

   a. In tab ‘Properties’ you will find the previously added component controller ‘CCallableObject1 – com.testcustomer.caf.eu.gp.ui.test’ as required controller.

   b. In tab ‘Actions’ create a new action ‘Complete’.
41. Change to the 'Layout' tab
   a. Design the UI that you require (you should be familiar with Web Dynpro development).
   b. Right click on RootUIElementContainer and choose Insert Child
   c. Insert Label, InputFields and bind the context elements to the values properties which are marked as follows:

   ![Label and InputFields](image)

   If you click on this line and on the context viewer will appear from where you can select the appropriate element (context mapping).

   **Note:**
   You may use the gridlayout for RootUIElementContainer and set colCount to “2”.
42. Add button ‘Complete’ and attach action ‘Complete’ to the properties of this button.

- With a click on the “Go” button you will move to the implementation part of the complete button (method onActionComplete).

43. In the ‘Implementation’ tab implement method ‘complete()’. You will call method ‘complete()’ of the component controller.

- First you need to implement the complete() method within the component controller.
Please add the Implementation for the onActionComplete Method as described.
45. Because we have implemented the interface IGPWebDynproCO we need to embed the view ‘VCallableObject1’ in window ‘WebDynproCO’.
   a. Right click window ‘WebDynproCO’
   b. Select **Embed View**
   c. Select **Embed exiting View**
   d. Click **Next**.
e. Select **VCallableObject1**
f. Click **Finish**.
46. If you like you can delete the Window WCCallableObject1. It will not be called in our application. The corresponding Interface View will be deleted too.

47. Build the development component and deploy it to your engine
   a. Select your project and right click
   b. Position on Development Component → Build…
   c. In the next window, click OK
   d. Check the View Tasks for errors.

Note:
On the right screen, the development components for gp/api and gp/api/wd were synced as projects and will be rebuilt too. Otherwise you will have only to build your local component.
48. Now deploy to your Engine.
   a. Select your project and right click
   b. Position on Development Component → Deploy
   c. Check the View Deploy Output View for errors.
   d. If you have problems with the deployment check your settings under Window → Preferences → SAP J2EE Engine. You may also check your running instance under the J2EE Development Perspective → View J2EE Engine.
Create an instance of the callable object

To create an instance of the callable object, please refer to the document: How To Create Callable Object Instances.doc. In the following section the Web Dynpro DC created will be used in a callable object. The callable object will be tested in the Design Time Environment without using an action, etc.
49. Open your Design Time within the portal: e.g. [http://localhost:53000/irj/index.html](http://localhost:53000/irj/index.html)
   
a. Navigate to tab “Guided Procedures”
   
b. Click on “Design Time”
   
c. Create a new folder “implementation”
   
d. Click on the new folder
   
e. Click on “Create Callable Object”
   
f. Select **Web Dynpro Component (GP Interface)** to find the new development DC, which will be here, because we have implemented the interface `IWebDynproCO`
   
g. Now enter an appropriate text under Name and Description e.g. Time-Off wd component
   
h. Because we have selected the newly created folder, it is already set under Location otherwise you can use the Choose... button to select your folder or even create a new one
   
i. Click on **Next**.
50. Now select `com.testcustomer.caf.eu.ui.test.CCallableObject1`
   
a. Click Next
   Now you can see the Input Parameters which we defined in `getDescription()`
   
b. Click Next
   Now you can see the Output Parameters which we defined in `getDescription()`
   
c. Click Next
   Now we are one step before testing.
   
d. Click Finish and Open.
51. Click on the **Test** tab
   
a. Enter vacation for one month, for instance

b. Click **Execute**

In the next screen you can define the pre-settings for the Input Parameters

- Now the execute() method within the interface controller will be called which *transports* the Input Parameters to the context and our callable object will be displayed on the screen

- You can change the start date, etc. and click on **Complete**

- Now method complete() will be called within the Component Controller. Complete() will transfer the context parameters to the Output Parameters and will set the execution state to successful and complete.