Develop composite applications with SAP NetWeaver Composition Environment 7.1

Part 3 — Modeling collaborative business processes with SAP tools

by Volker Stiehl

Volker Stiehl
Product Manager,
SAP AG

Volker Stiehl received a degree in computer science from the Friedrich-Alexander-University of Erlangen-Nürnberg, Germany. He was a consultant for distributed J2EE-based business solutions and integration architectures at Siemens for 12 years. In 2004 he joined SAP's product management team for a composite application development toolset, where he's also responsible for methodologies and best practices. He holds workshops in composite applications and is a regular speaker at conferences, such as SAPPHIRE, SAP TechEd, and JavaOne. You may reach him at volker.stiehl@sap.com.

Innovation — finding new ways to approach old problems — is the *raison d'être* of the business world, but innovation by itself is not the whole story. The speed with which you can change the way you approach a problem is equally as important. It's what makes your company competitive in the marketplace. Without the ability to change your approach quickly, your new solution loses much of its impact. It becomes just another entry into a crowded field. How can you speed up the process of getting your new and innovative solution to market? By using composite applications.

Composite applications are defined as packaged applications that sit on top of existing enterprise solutions reusing their functionality to form new, collaborative business processes. Composite applications also play an important role in SAP's enterprise service-oriented architecture (enterprise SOA) strategy. In fact, the announcement of SAP NetWeaver Composition Environment (SAP NetWeaver CE) in 2007 provided customers and partners with a design-time and runtime environment for composite applications for the first time. Since that announcement, the interest in composite applications and their development has grown dramatically.

This is the third and final installment in a series of articles that introduces you to the world of composite applications, their characteristics, their architecture, and the challenges you typically face developing them. The first installment (*SAP Professional Journal*, May/June 2008) introduces composite applications, SAP NetWeaver CE, and SAP Composite Application Framework (SAP CAF) and discusses the bottom layer of the composite application architecture shown in **Figure 1** on the next page: services and business objects. This first article also sets up the customer service complaint example.

The second installment (*SAP Professional Journal*, July/August 2008) delves into the next layer of the composite application architecture: user interfaces (UIs). It discusses the tool that you should use to create light-

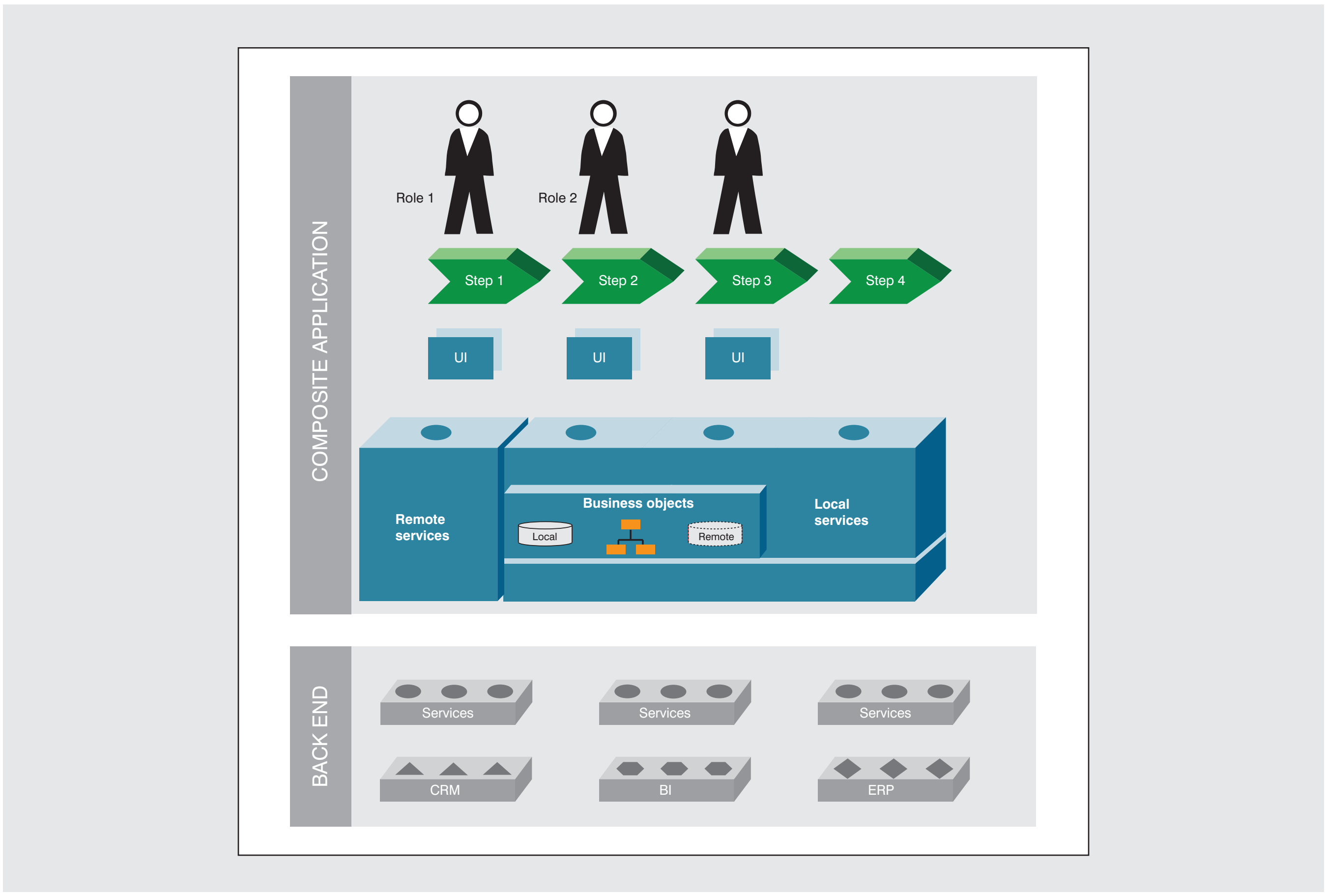


Figure 1 Architecture of a composite application

weight UIs: SAP NetWeaver Visual Composer. This installment also explains two techniques for consuming Web services and their advantages and disadvantages. It concludes by examining some advanced features available for composite applications.

This article, the third installment, explores the top layer of the composite application architecture: the process layer addressing the steps of the process with their assigned roles. This collaborative process modeling is carried out using SAP Guided Procedures (SAP GPs).

For this article, I assume that you have read the previous two articles in this series. This article builds upon that knowledge. As the previous articles have shown, you can break down a composite application development project into five steps:

Step 1: Search for services in SAP Enterprise Services Workplace (ES Workplace)

Step 2: Simplify the service interfaces with SAP CAF

Step 3: Create UIs with SAP NetWeaver Visual Composer

Step 4: Model business objects with SAP CAF

Step 5: Model collaborative processes with SAP GPs

Steps 1 and 2 were covered in Part 1 of this series, and Step 3 was discussed in Part 2. This article deals with Steps 4 and 5.

This step-by-step division helps to identify the different artifacts that make up a composite application: business objects, services, UIs, and processes.

Learn how to model processes with SAP GPs. As composite applications reuse existing functionality, part of this example is the consumption of real-world enterprise services provided by ES Workplace. Finally, you'll execute the process and learn how the SAP GP's framework helps you to navigate through the process steps.

Whether you're a Java developer or a business process expert looking for detailed information about composite applications and their development process, this article will give you a solid foundation from which to start your own journey into the world of composite applications.

Modeling business objects

Before coming to the process modeling part of the example, you have to think about the business objects on which the process is working. Let's recall the process flow of the example scenario: The last step is responsible for persisting the complaint data in a database. As SAP NetWeaver CE comes with its own database, it's possible to save application data there as well. Local persistency is necessary in cases where data isn't yet available for back-end systems. You can assume that a business object such as "complaint" isn't yet available, so you have to define the fields of this business object. Just by defining business objects, SAP CAF assumes all the cumbersome and error-prone tasks such as handling the data's persistency-inclusive database-locking, handling transactions, writing logs, and handling authorization and authentication.

Before you start modeling, think about your business object. Which fields are necessary to describe a complaint thoroughly? In most cases, you save the data that the process collected during execution. So far, the SAP GP has been responsible for persisting the data. Once the process ends, however, you lose all the collected data. As you want to run reports on complaints to improve quality, you must get access to this collected data. In essence, you collected information about the customer, the complaint, and the correspondence between customer, call center agent, and complaint manager.

It's not necessary to save all of the customer fields, such as name, city, or street. It's enough to save the customer's primary key (customerID) so that you can access the customer's details via an enterprise service call, if necessary. The customer address may change as well, so it's a good idea to call the enterprise service whenever you need it to fetch the latest customer details. You also need to know what types the fields are. First, consider which fields have their origin in a service call and which ones were newly defined for your scenario. Remember, you should always reuse the data types of those fields coming from the service calls. This leads you to the following list of fields for your complaint business object:

- customerID of type PartyIDContent (reused data type from the enterprise service)
- complaint of type LongText (new field)
- comment of type LongText (new field)
- approved of type Boolean

To reuse existing data types, you can import an enterprise service call directly into your SAP CAF project. With this technique you gain access to all of the data types used in the service call. However, if you want to separate your composite application into different SAP CAF projects, you must import the enterprise service into *every* SAP CAF project that needs access to it. If the interface to the enterprise service changes and you have to reread it, then each of the projects relying on that service has to reread the interface.

Therefore, the recommended approach is to create one SAP CAF project, which is solely responsible for the external data types (those used by enterprise services), and let other SAP CAF projects that need the data types simply reference this SAP CAF project. That way, the referencing SAP CAF projects don't have to import the interfaces as well. The following steps describe how to implement this recommendation:

1. Create a new SAP CAF project, as shown in Part 1 of this article, and name it spj_datastructs.
2. Right-click on your project's external node, and import the service customerBasicDataByID via the Services Registry (for details, see the section

"Step 2: Simplify service interfaces with CAF" in Part 1 of this series).

3. Create another SAP CAF project, and name it spj_complaint.

4. Open this spj_complaint project by double-clicking on its root node in the SAP Composite Application Explorer and the project's editor opens in the pane to the right of the Explorer.

5. The editor consists of two tabs: General and Reused projects. Switch to the Reused Projects tab, and click on the Add button. This enables you to set references to other SAP CAF projects, which saves effort because only one SAP CAF project has to import the interfaces.

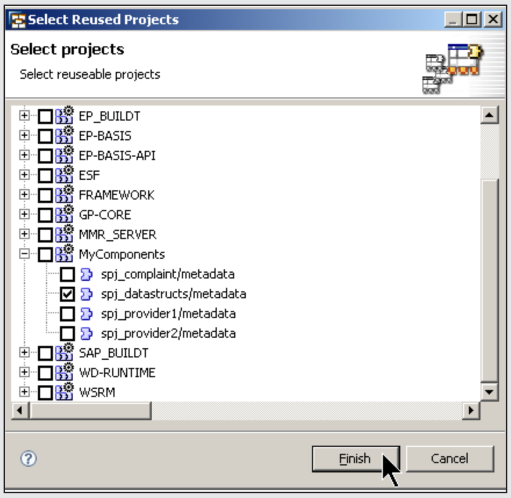
6. In the dialog that pops up, select Reused Projects, expand the MyComponents node, and set the checkmark for the spj_datastructs project. Click on Finish to set the reference (as shown in **Figure 2**).

The projects appear in the Composite Application Explorer, as shown in **Figure 3**. Now that this reference is set and the data types of the enterprise service are available for the business object, you can start the modeling process. Follow these steps:

1. Right-click on the modeled node of the spj_ complaint project, and choose New Business Object from the context menu. Set the new business object's name to Complaint and click on Finish. A new tab entitled Complaint opens, as shown in **Figure 4**.

2. Open the Structure tab at the bottom of the page to define the business object's data fields. SAP has already created some housekeeping fields for you, such as key (the business object's primary key), createdBy, createdAt, modifiedBy, and modifiedAt. SAP CAF fills them at runtime. You only have to fill the fields that you added to the business object.

3. Click on the Edit Main Structure button in the upper-left corner of the screen to open the attribute editor. On the left side of the editor's screen, you see the available existing data types from which you can choose; on the right side, you have a list of the attributes your business object contains.

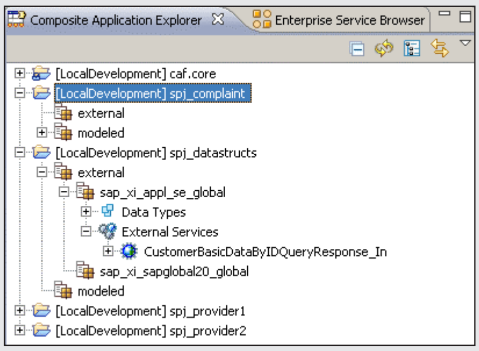

Select Reused Projects

Select projects

Select reuseable projects

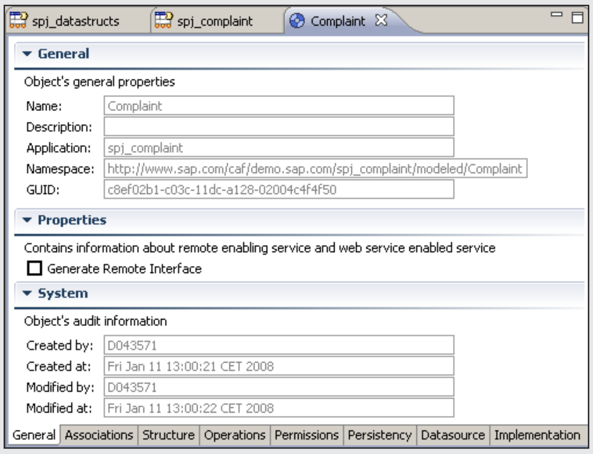
- ☐ EP_BUILDT
- ☐ EP-BASIS
- ☐ EP-BASIS-API
- ☐ ESF
- ☐ FRAMEWORK
- ☐ GP-CORE
- ☐ MMR_SERVER
- ☐ MyComponents
  - ☐ spj_complaint/metadata
  - ☑ spj_datastructs/metadata
  - ☐ spj_provider1/metadata
  - ☐ spj_provider2/metadata
- ☐ SAP_BUILDT
- ☐ WD-RUNTIME
- ☐ WSRM

Figure 2 Setting references to other projects



- [LocalDevelopment] caf.core
- [LocalDevelopment] spj_complaint
  - external
  - modeled
- [LocalDevelopment] spj_datastructs
  - external
    - sap_xi_appl_se_global
      - Data Types
      - External Services
        - CustomerBasicDataByIDQueryResponse_In
    - sap_xi_sapglobal20_global
  - modeled
- [LocalDevelopment] spj_provider1
- [LocalDevelopment] spj_provider2

Figure 3 The Composite Application Explorer



General

Object's general properties

Name:	Complaint
Description:	
Application:	spj_complaint
Namespace:	http://www.sap.com/caf/demo.sap.com/spj_complaint/modeled/Complaint
GUID:	c8ef02b1-c03c-11dc-a128-02004c4f4f50

Properties

Contains information about remote enabling service and web service enabled service

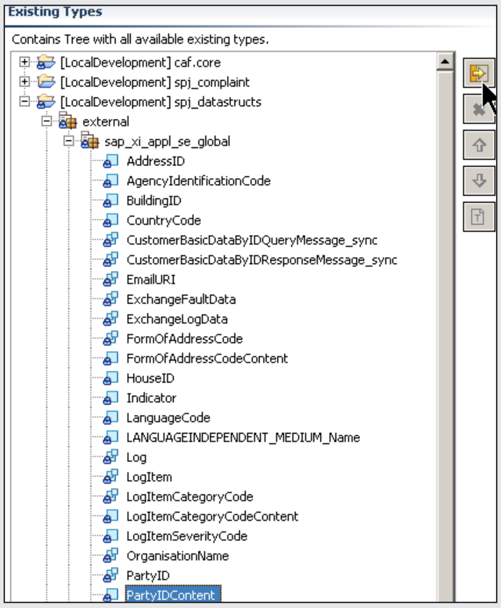
☐ Generate Remote Interface

System

Object's audit information

Created by:	D043571
Created at:	Fri Jan 11 13:00:21 CET 2008
Modified by:	D043571
Modified at:	Fri Jan 11 13:00:22 CET 2008

Figure 4 Creating a new business object — Complaint



Existing Types

Contains Tree with all available existing types.

```
[LocalDevelopment] caf.core
[LocalDevelopment] spj_complaint
[LocalDevelopment] spj_datastructs
    external
        sap_xi_appl_se_global
            AddressID
            AgencyIdentificationCode
            BuildingID
            CountryCode
            CustomerBasicDataByIDQueryMessage_sync
            CustomerBasicDataByIDResponseMessage_sync
            EmailURI
            ExchangeFaultData
            ExchangeLogData
            FormOfAddressCode
            FormOfAddressCodeContent
            HouseID
            Indicator
            LanguageCode
            LANGUAGEINDEPENDENT_MEDIUM_Name
            Log
            LogItem
            LogItemCategoryCode
            LogItemCategoryCodeContent
            LogItemSeverityCode
            OrganisationName
            PartyID
            PartyIDContent
```

Figure 5 Adding the customerID field as a data type

When you open the business object for the first time, this list contains only the housekeeping fields generated by SAP CAF.

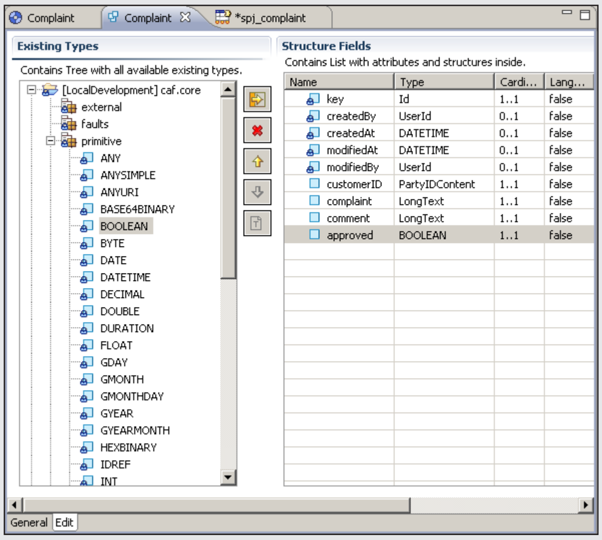
4. Let's start with the customerID field. Find the appropriate data type in the Existing Types section of the screen (on the left). The customerID has a data type of PartyIDContent (party as in customers, suppliers, etc.), and it's part of the external service. Therefore, you should expand the spj_datastructs → external → sap_xi_appl_se_global nodes, select PartyIDContent, and click on the Add button, as shown in **Figure 5**.

5. The field appears below the existing attributes ready for renaming. Name it customerID.

Important!

Give the same name to *all* of the fields in your composite application that contain the same information. I cannot stress this enough; its importance becomes obvious later in the SAP GP part of this article.

6. Repeat the assignment of fields (complaint, comment, and approved) so that your final business object looks like the one in **Figure 6**.



Existing Types

Contains Tree with all available existing types.

```
[LocalDevelopment] caf.core
    external
    faults
    primitive
        ANY
        ANYSIMPLE
        ANYURI
        BASE64BINARY
        BOOLEAN
        BYTE
        DATE
        DATETIME
        DECIMAL
        DOUBLE
        DURATION
        FLOAT
        GDAY
        GMONTH
        GMONTHDAY
        GYEAR
        GYEARMONTH
        HEXBINARY
        IDREF
        INT
```

Structure Fields

Contains List with attributes and structures inside.

Name	Type	Cardi...	Lang...
key	Id	1..1	false
createdBy	UserId	0..1	false
createdAt	DATETIME	0..1	false
modifiedAt	DATETIME	0..1	false
modifiedBy	UserId	0..1	false
customerID	PartyIDContent	1..1	false
complaint	LongText	1..1	false
comment	LongText	1..1	false
approved	BOOLEAN	1..1	false

Figure 6 Defining the Complaint business object

Important!

All of the new fields — customerID, complaint, comment, and approved — have 1:1 cardinality. This means that those fields are mandatory now, and you must fill them each time a new business object of that type is created. This cardinality also tells SAP CAF that the method for creating business objects needs to use those fields as input parameters. Adapt the drop-down field in the Cardinality column accordingly by simply clicking into the field and choosing 1..1 from the list. Believe it or not, that's all you have to do to model a business object with its own persistency.

However, it is recommended that you also look at the other tabs (see **Figure 7**):

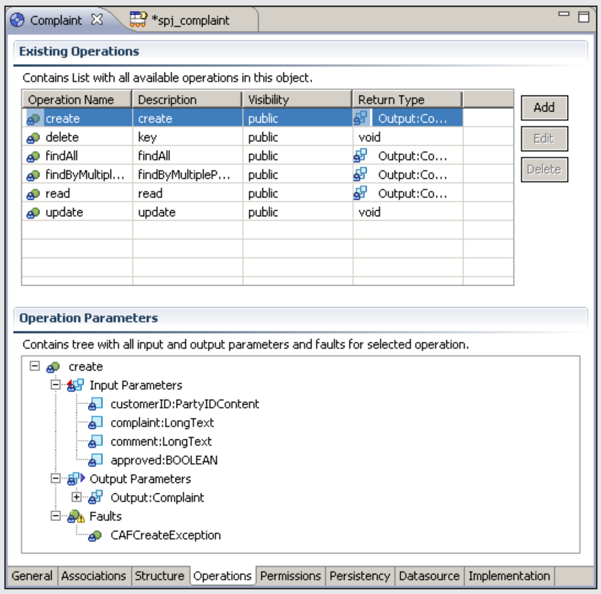
- On the Associations tab, you can model relationships between business objects, for example, one-to-many, many-to-one, many-to-many, and even bidirectional relationships, to name just a few. Behind the scenes, SAP CAF takes care of transactional integrity.
- On the Operations tab (shown in detail), you get a list of all generated life-cycle methods for your business object.

This screen shows you another consequence of setting the field's cardinality field to 1:1. The selected Create method has input parameters, which wouldn't be the case if you had left the initial settings for the fields. In addition to the methods listed, you can add more finder methods (search methods for which the names begin with "find"), depending on the needs of your composite application.

For example, if you need to search for those complaints that contain a certain text within their comment field, then it's a good idea to add a findByComment search method. To name it, I recommend you start with findBy followed by the fields on which the finder relies. However, you can assign any name you want. Just click on the Add button, pick the fields for your finder method's input parameters, and provide an appropriate name. That's it! SAP CAF automatically covers the rest, especially implementing the search functionality.

- On the Permissions tab, you define whether you need access rules for your business object. On this tab, you simply enable or disable permission checks. The access rules themselves are defined outside the SAP NetWeaver Developer Studio (NWDS) with a configuration tool that comes with SAP CAF. Simply put, just remove the checkmark from the Permission checks enabled checkbox.
- On the Persistency tab, you switch between local and remote persistency. You can also determine which database tables will be created based on your business object's fields. The database table being created for the complaint looks like the one shown in **Figure 8**.

To access the Create method from the process layer, you must make it accessible via a Web service call. Here again, SAP CAF can help you tremendously: Simply right-click on your business object



Existing Operations

Contains List with all available operations in this object.

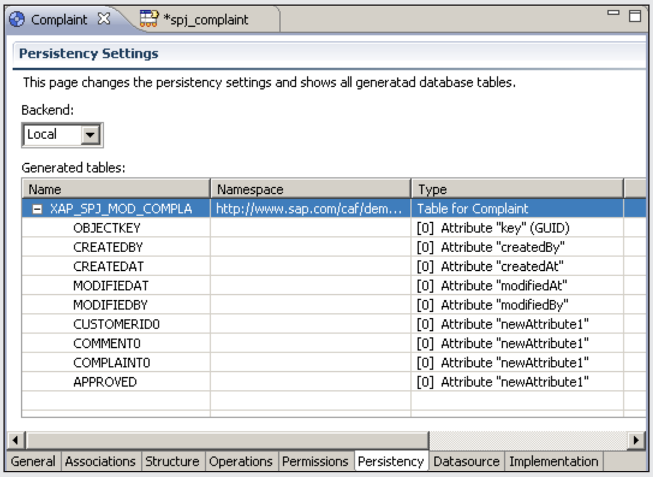
Operation Name	Description	Visibility	Return Type
create	create	public	Output:Co...
delete	key	public	void
findAll	findAll	public	Output:Co...
findByMultipl...	findByMultipleP...	public	Output:Co...
read	read	public	Output:Co...
update	update	public	void

Operation Parameters

Contains tree with all input and output parameters and faults for selected operation.

```
create
  Input Parameters
    customerID:PartyIDContent
    complaint:LongText
    comment:LongText
    approved:BOOLEAN
  Output Parameters
    Output:Complaint
  Faults
    CAFCreateException
```

Figure 7 Life-cycle methods of the Complaint business object



Complaint | *spj_complaint

Persistency Settings

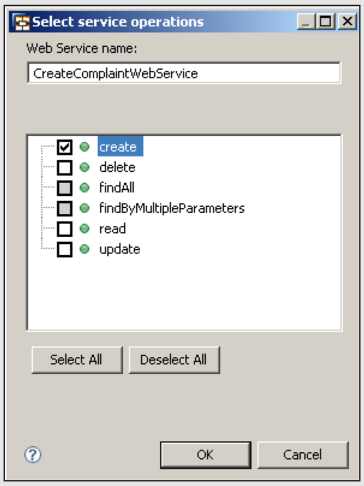
This page changes the persistency settings and shows all generatad database tables.

Backend: Local

Generated tables:

Name	Namespace	Type
XAP_SPJ_MOD_COMPLA	http://www.sap.com/caf/dem...	Table for Complaint
OBJECTKEY		[0] Attribute "key" (GUID)
CREATEDBY		[0] Attribute "createdBy"
CREATEDAT		[0] Attribute "createdAt"
MODIFIEDAT		[0] Attribute "modifiedAt"
MODIFIEDBY		[0] Attribute "modifiedBy"
CUSTOMERID0		[0] Attribute "newAttribute1"
COMMENT0		[0] Attribute "newAttribute1"
COMPLAINT0		[0] Attribute "newAttribute1"
APPROVED		[0] Attribute "newAttribute1"

Figure 8 Associated database table for Complaint business object



Select service operations

Web Service name: CreateComplaintWebService

- [✓] create
- [ ] delete
- [ ] findAll
- [ ] findByMultipleParameters
- [ ] read
- [ ] update

Figure 9 Exposing life-cycle methods as Web services

node in SAP Composite Application Explorer, and select Expose service as Web Service from the context menu. In the dialog that pops up, select the service you want to enable as a Web service and provide an appropriate name. For the example, you only need the Create method; use CreateComplaintWebService to name the service (see **Figure 9**) and click on OK.

Generate, build, and deploy

With the Web service now in place, you can generate, build, and deploy your business object. Right-click on the spj_complaint project node in SAP Composite Application Explorer, and execute the following:

- Generate pure Java code for your business object
- Translate the code into Java binaries and pack them into deployable Java archives
- Deploy the archive to the Java server on which the application will run

Your newly created business object is now ready for testing.

Right-click on the business object Complaint in the Composite Application Explorer and click on Test Service from the context menu. The Service Browser (explained in Part 1) opens in a new browser window. Drill down in the tree at the left side of the window until the business object Complaint appears and select it.

To test the persistency functionality, click on the New button (a new empty row appears in the table representing the contents of the database) and enter data only for those fields of your business object that you've added to it (customerID, complaint, comment, approved). Click on Save to call the complaint's Create method (if it's a new entry) or the Update method (if you're modifying an existing entry).

Each of the buttons in the Data Component pane connects to the life-cycle methods of the business object. Clicking on a button generates the primary key and automatically fills the bookkeeping fields (e.g., createdBy, createdAt) to indicate that the method has successfully created the new Complaint business object in the database table (see **Figure 10** for details).

This table lists all instances of existing business objects. The findAll method (under Available Services on the left) conveniently lists all database entries; one database entry reflects one Complaint business object. To have local persistency in your composite application, you need to save all created complaints for later analysis.

To consume the Create method on the process layer using a Web service call, SAP recommends that you check the generated Web service as well. To do this, you use the SAP NetWeaver Web Services Navigator. When the Web service successfully calls the business object, you see the results in a format like that in **Figure 11**.

With the Create Web service for the Complaint business object now in place, you need to create a logical destination for the Web service. You want to clearly decouple the service consumers from the service providers (as explained in Part 1) by using logical destinations. Therefore, create the logical destination CafComplaintWebServiceDest in SAP NetWeaver Administrator (NWA), and assign your newly created Web service to it. Here, the business object-modeling exercise ends.

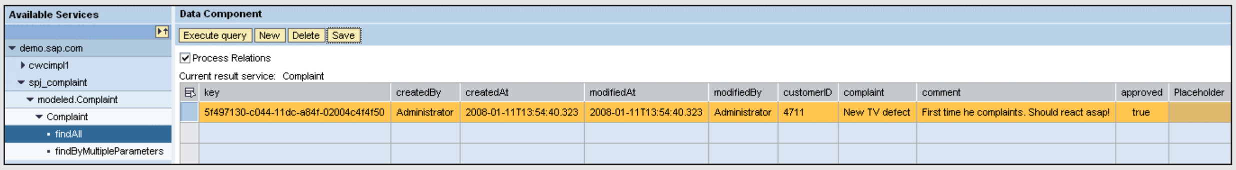
Now, let's review what you've done in this exercise with just a few mouse clicks: You've created a business object for which you produced a database table, generated life-cycle methods, provided a Web service, and defined access control. For this entire sequence, SAP CAF covers the database-handling, inclusive, database entry-locking and transaction management. Imagine that you had to build all of this functionality manually with other technologies, such as pure Java. This relative simplicity is the real value of model-driven development and code generation.

In the previous steps, you've created all of the artifacts you need for your composite application. Now you glue them together into a collaborative process using SAP GPs.

Modeling collaborative processes

SAP GP is the tool of choice for modeling workflow processes that involve human interactions as well as background steps, which call services in back-end systems. SAP GP is a complete framework for modeling and managing processes and is designed to implement business processes with greater ease and speed across multiple applications. Although it lacks a graphical design time, the framework enables even those users without specialized software development skills to easily set up and execute collaborative processes.

During runtime the end users receive notification of their participation in a process in which the participants fill out either online forms or interactive offline forms based on Adobe technology. This is how SAP GP navigates (guides) the participants through the process. It's a push mechanism, and the notifications

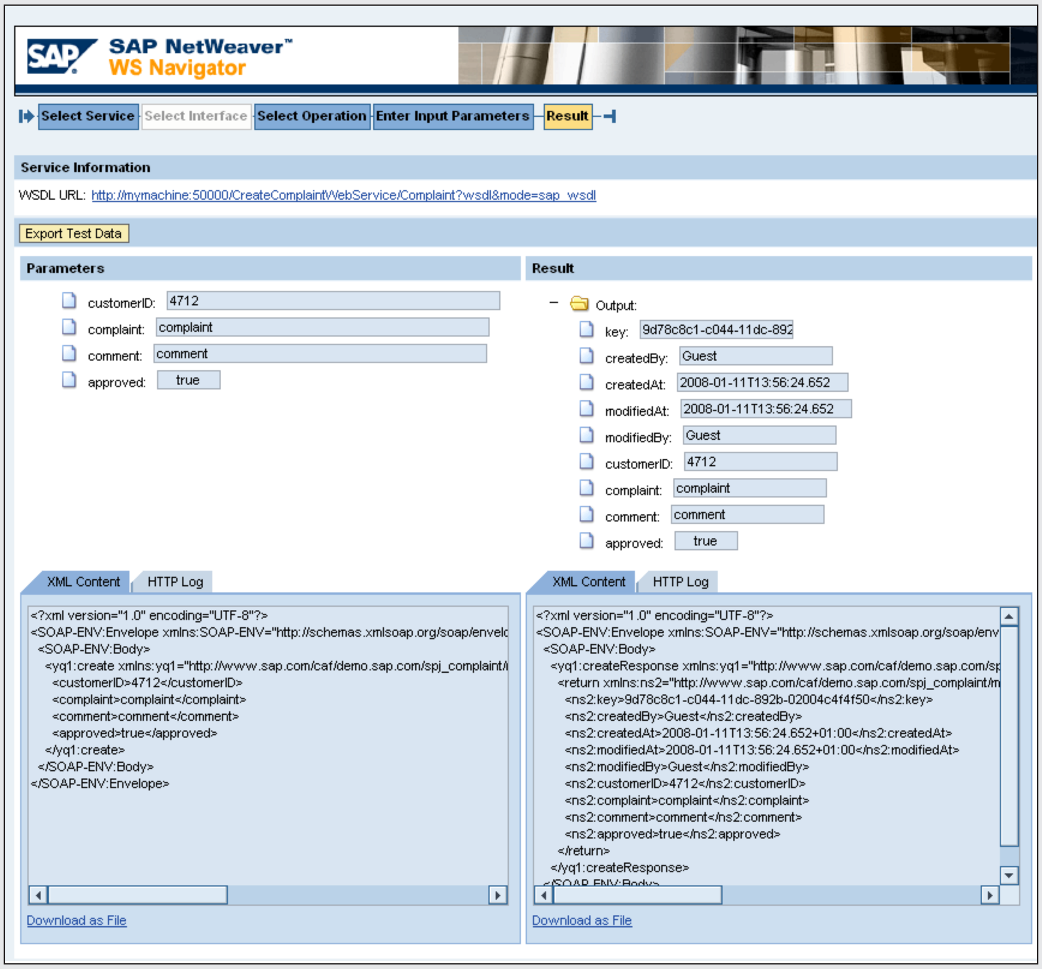


Process Relations

Current result service: Complaint

key	createdBy	createdAt	modifiedAt	modifiedBy	customerID	complaint	comment	approved	Placeholder
5f497130-c044-11dc-a84f-02004c4f4f50	Administrator	2008-01-11T13:54:40.323	2008-01-11T13:54:40.323	Administrator	4711	New TV defect	First time he complaints. Should react asap!	true	

Figure 10 Creating an entry in the database table



Service Information

WSDL URL: http://mymachine:50000/CreateComplaintWebService/Complaint?wsdl&mode=sap_wsdl

Parameters

customerID:	4712
complaint:	complaint
comment:	comment
approved:	true

Result

Output:

key:	9d78c8c1-c044-11dc-892
createdBy:	Guest
createdAt:	2008-01-11T13:56:24.652
modifiedAt:	2008-01-11T13:56:24.652
modifiedBy:	Guest
customerID:	4712
complaint:	complaint
comment:	comment
approved:	true

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelo
 <SOAP-ENV:Body>
  <yq1:create xmlns:yq1="http://www.sap.com/caf/demo.sap.com/spj_complaint/
   <customerID>4712</customerID>
   <complaint>complaint</complaint>
   <comment>comment</comment>
   <approved>true</approved>
  </yq1:create>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/env
 <SOAP-ENV:Body>
  <yq1:createResponse xmlns:yq1="http://www.sap.com/caf/demo.sap.com/sp
   <return xmlns:ns2="http://www.sap.com/caf/demo.sap.com/spj_complaint/m
    <ns2:key>9d78c8c1-c044-11dc-892b-02004c4f4f50</ns2:key>
    <ns2:createdBy>Guest</ns2:createdBy>
    <ns2:createdAt>2008-01-11T13:56:24.652+01:00</ns2:createdAt>
    <ns2:modifiedAt>2008-01-11T13:56:24.652+01:00</ns2:modifiedAt>
    <ns2:modifiedBy>Guest</ns2:modifiedBy>
    <ns2:customerID>4712</ns2:customerID>
    <ns2:complaint>complaint</ns2:complaint>
    <ns2:comment>comment</ns2:comment>
    <ns2:approved>true</ns2:approved>
   </return>
  </yq1:createResponse>
 </SOAP-ENV:Body>
```

Figure 11 Calling the Create method using a Web service

contain links to the screen that the end user has to fill out. So the end user is actually guided. That's why it's called *Guided* Procedures.

In contrast to the online scenarios, SAP GP also supports offline scenarios. Here, an offline interactive form based on Adobe technology (e.g., a PDF form) takes over the interactive step. You can store the PDF locally on an end user's notebook: the user can fill it out whenever he or she wants, and then transfer it back to continue the process when he or she is online again.

In essence, SAP GP consists of three major parts:

- **Design time:** In the SAP GP design time, you primarily decide upon the process flow, assign concrete executable applications to each step of the process, manage the data flow between steps, and assign process roles to each step. In the end you model a process template, which can be instantiated via a simple Web URL, a Web service call, or a Java API.

- **Runtime:** During the instantiation of the process, you assign concrete users or groups to the various process roles so that the SAP GP runtime knows to whom to send notifications. These notifications are work items that show up in the user's

Universal Worklist (UWL) inbox. From there, the user can navigate to the appropriate step of the procedure.

- **Administration:** You need SAP GP administration to maintain processes (e.g., terminate a process), maintain the email templates you need to circulate PDF forms, schedule process instantiation, or transport processes, just to name a few functions.

Now, let's develop the process flow for the example scenario (described in Part 1).

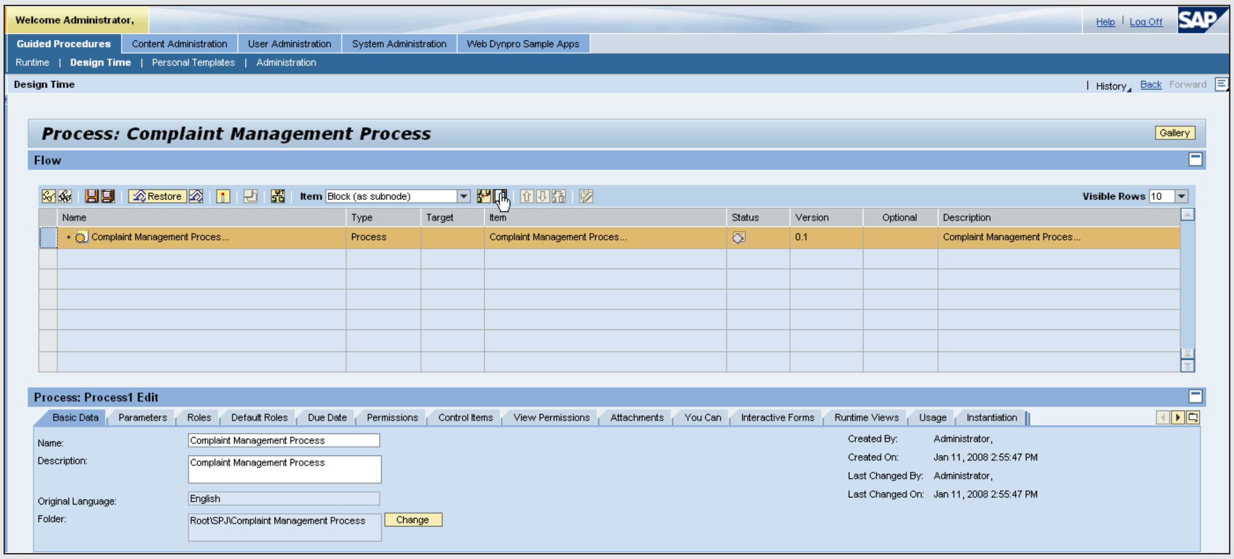
Process flow

You need to model four steps, which execute sequentially with one jump back to the review step if the complaint manager has further questions. Start the portal via the URL http://<*host*>:<*port*>/irj, log on to the portal, and, by first- and second-level navigation, go to the SAP GP's design time. If you don't see the tabs for SAP GPs, you need to assign SAP GP roles to your user account. Details about the roles you need to work with SAP GP can be found on the SAP Help Portal (http://help.sap.com/ → SAP NetWeaver → SAP NetWeaver CE → SAP NetWeaver Composition Environment Library → Administrator's Guide → Configuration of SAP NetWeaver CE → Configuration for CE Additional Components → Configuring Guided Procedures → Setting Up Portal Roles).

If you need a more thorough introduction to SAP GPs, I recommend online help for SAP NetWeaver CE. I also recommend the basic tutorial, "Developing Your First Process," on the SAP Help Portal (http://help.sap.com → SAP NetWeaver → SAP NetWeaver CE → SAP NetWeaver Composition Environment Library → Developer's Guide → Developing and Composing Applications → Designing Composite Processes with Guided Procedures → Reference → Tutorials) for a taste of the environment.

Create a new process in a folder of your choice, and name it Complaint Management Process (see **Figure 12** for details). Underneath the process node, place a sequential block with the same name as the process node (in this case, Complaint Management Process). The block itself contains five actions representing the different activities that the process must execute. They are:

- Enter Complaint
- Approve Complaint



Process: Complaint Management Process

Name	Type	Target	Item	Status	Version	Optional	Description
Complaint Management Proces...	Process		Complaint Management Proces...		0.1		Complaint Management Proces...

Process: Process1 Edit

Name:	Complaint Management Process
Description:	Complaint Management Process
Original Language:	English
Folder:	Root\SPJ\Complaint Management Process
Created By:	Administrator,
Created On:	Jan 11, 2008 2:55:47 PM
Last Changed By:	Administrator,
Last Changed On:	Jan 11, 2008 2:55:47 PM

Figure 12 Newly created process in the SAP GP design time

- Review Complaint
- Jump to Approval
- Create Complaint

New to the initial process (described in Part 1) is the Jump to Approval action. SAP GP has no graphical process flow development environment. As the block is sequential, all five of these actions execute one after another. Jumps are only possible if a UI fires a result state or if a dedicated jump action occurs. I use both options in this series of articles. Remember from the UI modeling exercise in SAP NetWeaver Visual Composer (described in Part 2), the Approve Complaint action can signal one of two result states: Finish or Review. If the option is Finish, you jump directly to the final action, Create Complaint. If the option is Review, you don't jump immediately. Instead, the process continues with the Review Complaint action. Then, it executes the Jump to Approval action, which actually jumps back to the Approve Complaint action.

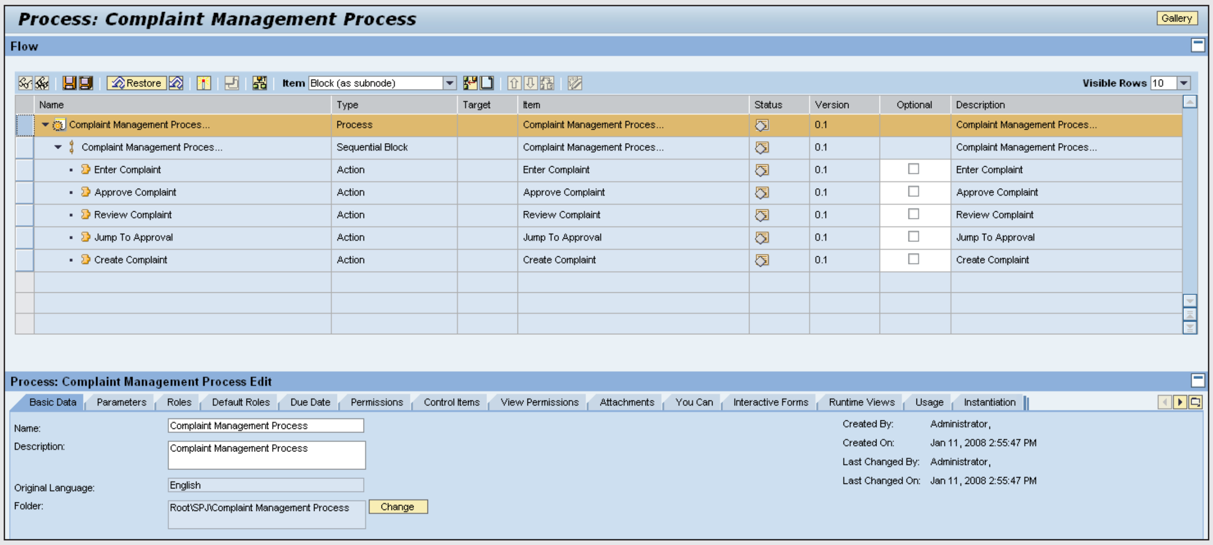
After you've entered the actions, the flow looks like the one shown in **Figure 13**.

Each action functions as a placeholder for either an interactive step or a background call. The next task is to assign the appropriate callable object to each of the actions. A *callable object* is an SAP GP-related term for something executable, either in a dialog or in the background.

Enter Complaint

Assign a callable object of type Web Dynpro for Visual Composer (WD4VC). **Figure 14** on the next page shows you several UIs you can plug into an action. However, for a tight integration between SAP GP and a UI, SAP recommends only the first two entries: Web Dynpro Component (GP Interface) and WD4VC Application. This is because only those UIs can receive parameters from SAP GP *and* return parameters to SAP GP at the same time. You can either call *other* UI technologies without any data transfer or with data transfer from SAP GP to the UI (but not vice versa). Therefore, if you want a smooth integration between the UI and SAP GP, stick with Web Dynpro Component (GP Interface) and WD4VC Application.

It's also a good habit to name the callable object the same as the action to which it is assigned. Click



Name	Type	Target	Item	Status	Version	Optional	Description
Complaint Management Proces...	Process		Complaint Management Proces...		0.1		Complaint Management Proces...
Complaint Management Proces...	Sequential Block		Complaint Management Proces...		0.1		Complaint Management Proces...
Enter Complaint	Action		Enter Complaint		0.1	☐	Enter Complaint
Approve Complaint	Action		Approve Complaint		0.1	☐	Approve Complaint
Review Complaint	Action		Review Complaint		0.1	☐	Review Complaint
Jump To Approval	Action		Jump To Approval		0.1	☐	Jump To Approval
Create Complaint	Action		Create Complaint		0.1	☐	Create Complaint

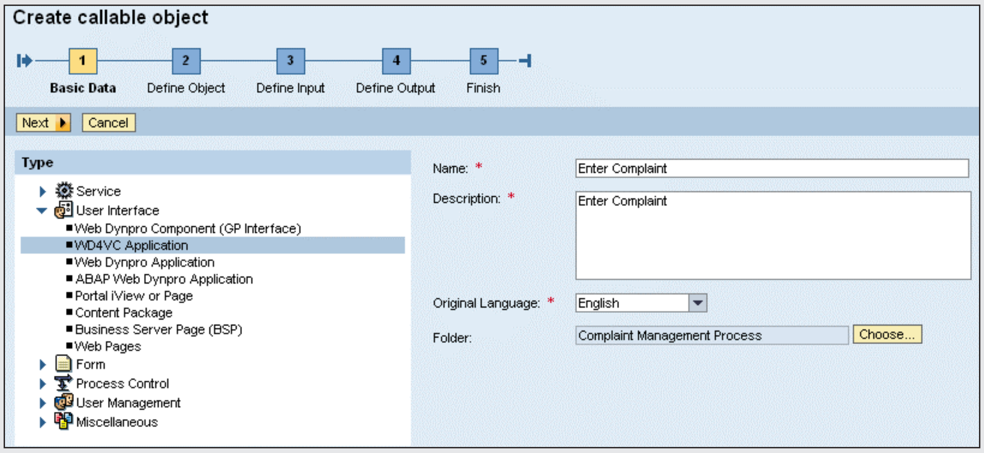
Name: Complaint Management Process
Description: Complaint Management Process
Original Language: English
Folder: Root\SPJ\Complaint Management Process

Created By: Administrator,
Created On: Jan 11, 2008 2:55:47 PM
Last Changed By: Administrator,
Last Changed On: Jan 11, 2008 2:55:47 PM

Figure 13 Process flow after adding the five actions

on Next to choose the appropriate UI, as shown in **Figure 15**.

Because of your selection of WD4VC as a filter criterion (**Figure 14**), SAP NetWeaver Visual Composer UIs are listed here. Use the filter capability of this dialog (below and to the right of the funnel icon), especially if you have a large number of UIs. Finish the wizard that creates the callable object by walking through all of the wizard screens. After finishing the creation of the SAP NetWeaver Visual Composer callable object, the process flow editor looks like the one shown in **Figure 16**.



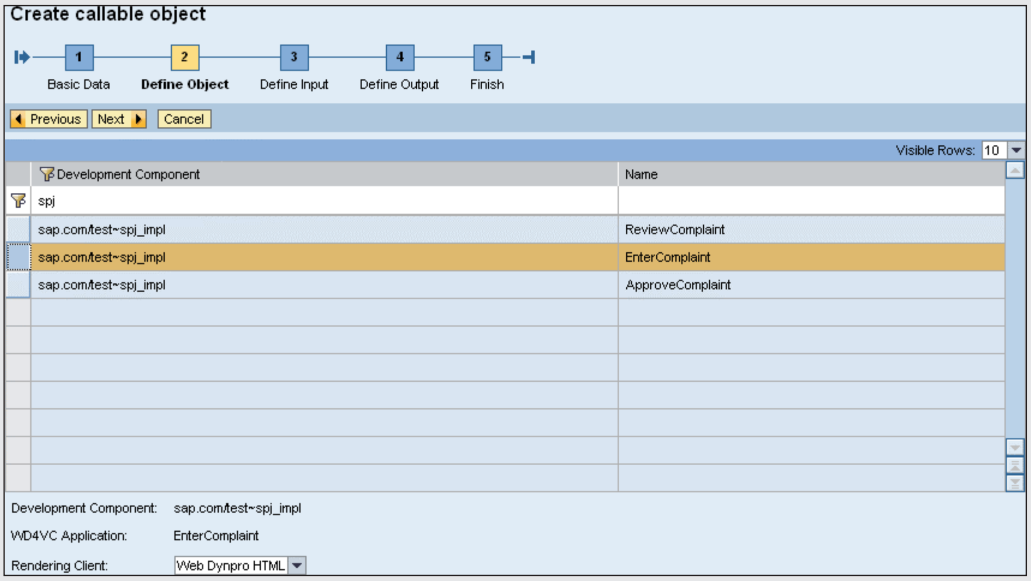
Name: * Enter Complaint

Description: * Enter Complaint

Original Language: * English

Folder: Complaint Management Process

Figure 14 Creating a callable object of type WD4VC



Development Component	Name
spj	
sap.com/test~spj_impl	ReviewComplaint
sap.com/test~spj_impl	EnterComplaint
sap.com/test~spj_impl	ApproveComplaint

Development Component: sap.com/test~spj_impl

WD4VC Application: EnterComplaint

Rendering Client: Web Dynpro HTML

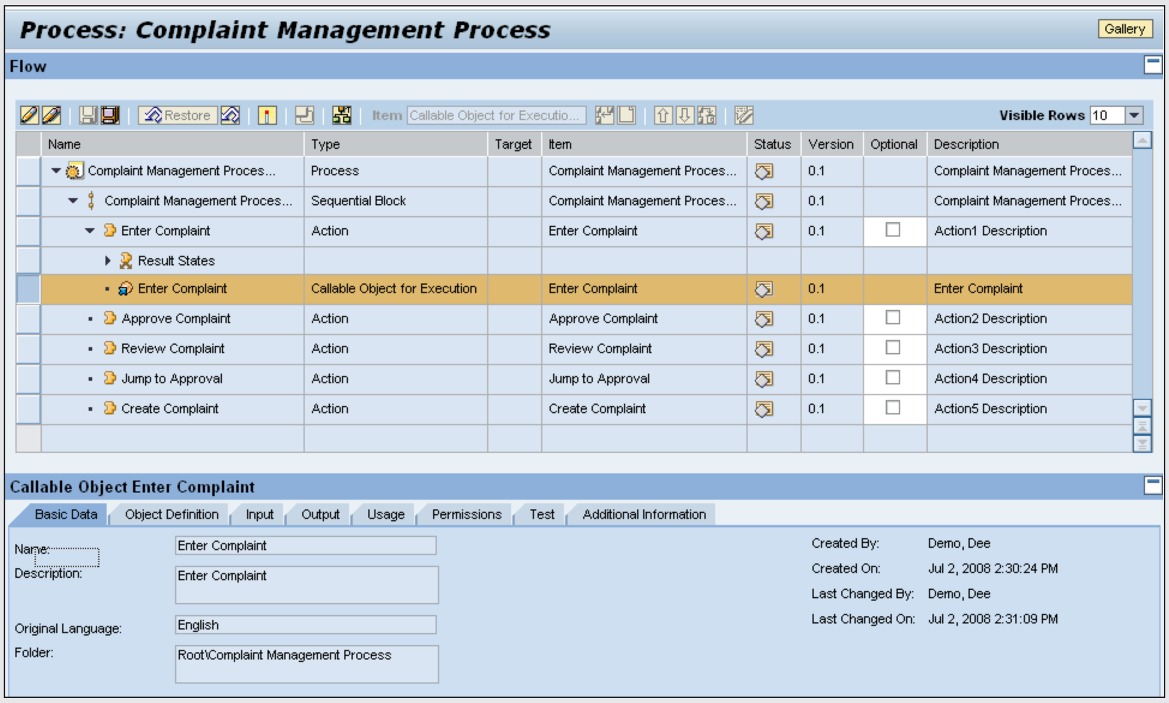
Figure 15 Assigning an SAP NetWeaver Visual Composer UI to a callable object

Next, it's important to test your UI in the SAP GP environment. The SAP GP design time enables you to test each callable object individually and shows you how it interacts with the rest of SAP GP. For example, you can figure out whether the data transfers correctly to the UI or is handed back from the UI to SAP GP and whether the result states fire correctly. Only Web Dynpro and SAP NetWeaver Visual Composer UIs can exchange data with SAP GP in both directions.

In the process flow editor, select the row of the newly created callable objects. Beneath the table, you find several tabs that allow you to configure the callable object appropriately. One tab is named Test. Once you select it, a wizard starts.

In step 1 of the wizard, you prepare some input data for your UI. SAP recommends that you always make use of this feature. By doing this, you ensure proper behavior during runtime. However, the Enter Complaint step doesn't require any inputs, so you proceed to step 2 of the wizard by clicking on Execute, which shows the SAP NetWeaver Visual Composer UI as if it would run in the SAP GP runtime. Play with the UI to see that the service calls work. Fill the input fields and click on Send to Complaint Manager on the modeled SAP NetWeaver Visual Composer screen to leave the Enter Complaint test screen. At the same time, you reach the last step of the test wizard. Here, you can analyze the return values, as shown in **Figure 17** on the next page.

First, make sure that the execution completes successfully. Next, analyze the result state. In this case you didn't model a result state; therefore, the system displays the default result state. Finally, check the returned parameters. I highly recommend that you *always* check your callable objects this way. It saves you a lot of time if you run into problems while executing your complete process, and you know for certain whether the individual callable objects behave as expected.



Process: Complaint Management Process

Flow

Name	Type	Target	Item	Status	Version	Optional	Description
Complaint Management Proces...	Process		Complaint Management Proces...		0.1		Complaint Management Proces...
Complaint Management Proces...	Sequential Block		Complaint Management Proces...		0.1		Complaint Management Proces...
Enter Complaint	Action		Enter Complaint		0.1	☐	Action1 Description
Result States							
Enter Complaint	Callable Object for Execution		Enter Complaint		0.1		Enter Complaint
Approve Complaint	Action		Approve Complaint		0.1	☐	Action2 Description
Review Complaint	Action		Review Complaint		0.1	☐	Action3 Description
Jump to Approval	Action		Jump to Approval		0.1	☐	Action4 Description
Create Complaint	Action		Create Complaint		0.1	☐	Action5 Description

Callable Object Enter Complaint

Basic Data | Object Definition | Input | Output | Usage | Permissions | Test | Additional Information

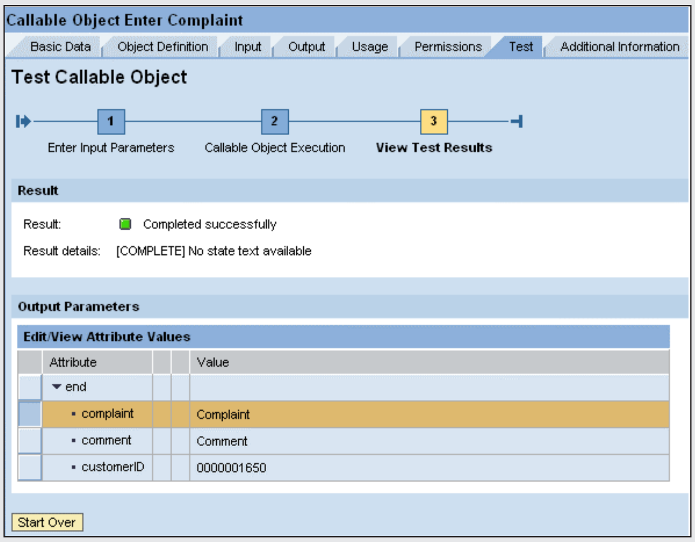
Name:	Enter Complaint
Description:	Enter Complaint
Original Language:	English
Folder:	Root\Complaint Management Process
Created By:	Demo, Dee
Created On:	Jul 2, 2008 2:30:24 PM
Last Changed By:	Demo, Dee
Last Changed On:	Jul 2, 2008 2:31:09 PM

Figure 16 Process flow editor after assigning a callable object to Enter Complaint

Approve Complaint

Repeat the assignment of a callable object to an action for the Approval step. It's a type WD4VC callable object, so just follow the steps described in the "Enter Complaint" section on page 51. Now, you want to essentially repeat this process for the second UI (Approve Complaint), but with two cautions:

- Because the Approval UI contains result states that depend on the button pressed, test both buttons (Finished and Return to CC) and double-check the result states to see whether the result state you modeled in Part 1 fired successfully after execution. For example, look at **Figure 18**.



Result

Result: 🟩 Completed successfully

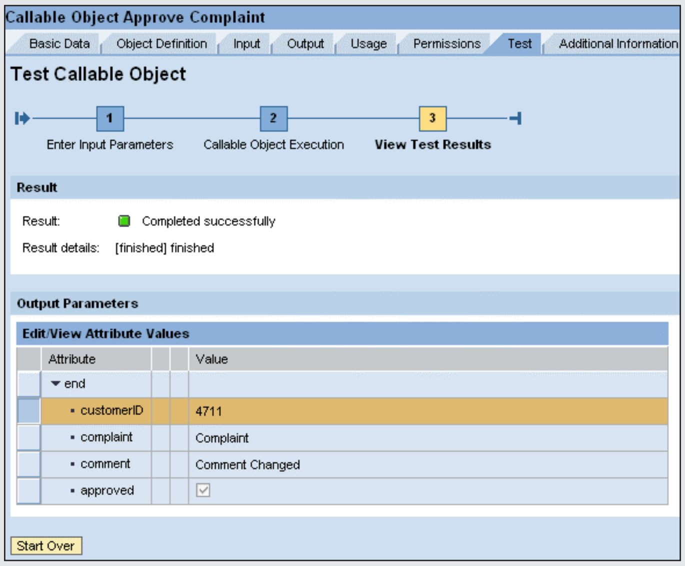
Result details: [COMPLETE] No state text available

Output Parameters

Edit/View Attribute Values

Attribute	Value
▾ end	
• complaint	Complaint
• comment	Comment
• customerID	0000001650

Figure 17 Analyzing the return values of the callable object test



Result

Result: 🟩 Completed successfully

Result details: [finished] finished

Output Parameters

Edit/View Attribute Values

Attribute	Value
▾ end	
• customerID	4711
• complaint	Complaint
• comment	Comment Changed
• approved	✓

Figure 18 Correct result state for the Approval screen

- Make sure that both result states appear in the process flow editor as well. In addition, model a jump target for the Finished result state. Remember, you have to jump to the Create Complaint action in case the user clicked on the Finished button (see **Figure 19**).

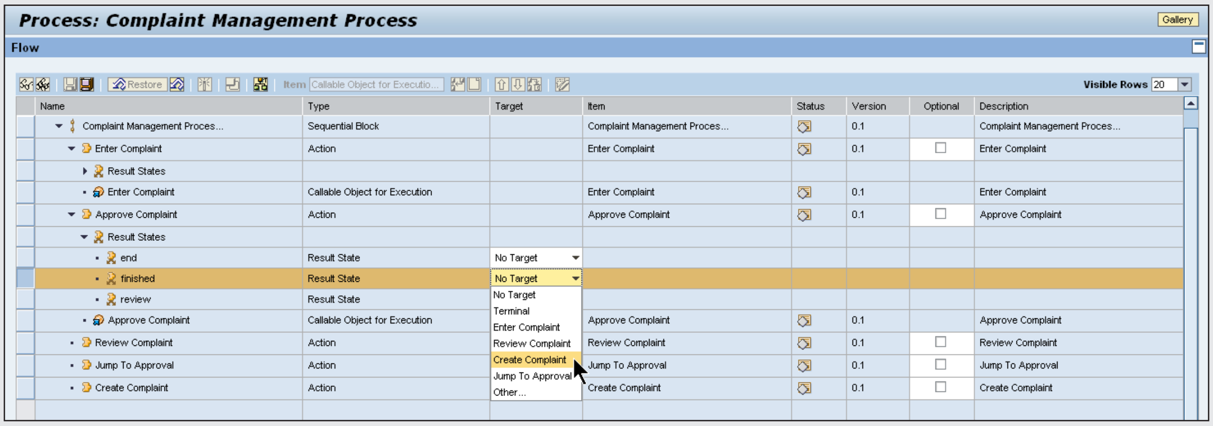
Review Complaint

With this knowledge, you are now prepared to add the third screen to the Review Complaint screen and finish the UI part of the process.

Jump to Approval

Next, you move on to the Jump to Approval action. To assign a jump step to the process, you make use of another predefined callable object type; it's named Decision (Comparison to Predefined Value) and you can find it underneath the Process Control node, as shown in **Figure 20**.

This callable object consists of one input parameter which is then compared to the predefined fixed value you are going to define next. Depending on the outcome of this comparison, the callable object fires one of two



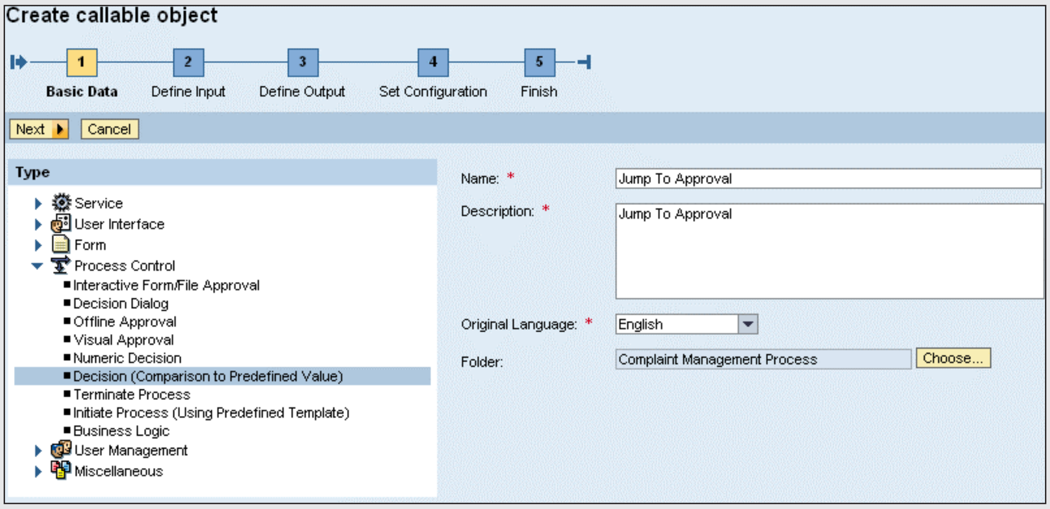
Process: Complaint Management Process

Flow

Visible Rows 20

Name	Type	Target	Item	Status	Version	Optional	Description
Complaint Management Proces...	Sequential Block		Complaint Management Proces...	◇	0.1		Complaint Management Proces...
Enter Complaint	Action		Enter Complaint	◇	0.1	☐	Enter Complaint
Result States							
Enter Complaint	Callable Object for Execution		Enter Complaint	◇	0.1		Enter Complaint
Approve Complaint	Action		Approve Complaint	◇	0.1	☐	Approve Complaint
Result States							
end	Result State	No Target					
finished	Result State	No Target					
review	Result State						
Approve Complaint	Callable Object for Execution		Approve Complaint	◇	0.1		Approve Complaint
Review Complaint	Action		Review Complaint	◇	0.1	☐	Review Complaint
Jump To Approval	Action		Jump To Approval	◇	0.1	☐	Jump To Approval
Create Complaint	Action		Create Complaint	◇	0.1	☐	Create Complaint

Figure 19 Defining a jump target for a result state



Create callable object

Type
- Service
- User Interface
- Form
- Process Control
  - Interactive Form/File Approval
  - Decision Dialog
  - Offline Approval
  - Visual Approval
  - Numeric Decision
  - Decision (Comparison to Predefined Value)
  - Terminate Process
  - Initiate Process (Using Predefined Template)
  - Business Logic
- User Management
- Miscellaneous

Name: *	Jump To Approval
Description: *	Jump To Approval
Original Language: *	English
Folder:	Complaint Management Process

Figure 20 Callable object type decision

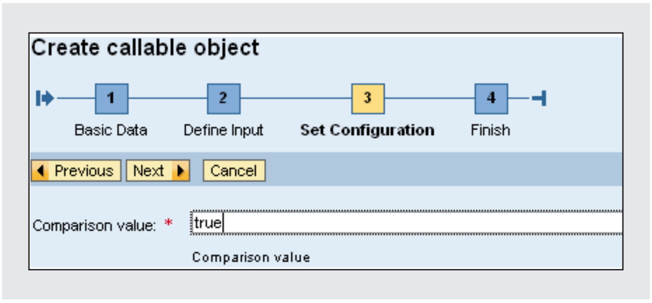
possible result states: either equal or different. The callable object's input parameter has the name Input parameter as you can verify with the Define Input step of the wizard. Click on Next until you reach the Set Configuration step. Here, you define the comparison value to which to compare the input parameter. You can use any value you like. In the example, I used true as the comparison value (see **Figure 21**).

That's all you need to do to define the callable object.

So far you have only defined the comparison value but how do you make sure that the process really jumps back? Well, this brings me to the next feature of SAP GPs I'd like to explain: default values. It's possible to assign default values to certain parameters. Obviously, you can do this for the newly created callable object as well. It has only one input parameter, and I'd like to set it to exactly the comparison value above. By doing this, you make sure that the outcome of the comparison is *always* equal. Therefore, select the action in the process flow editor to which your callable object is assigned (default values can only be set on the action level), and select the Parameters tab on the lower part of the screen. Select the only input parameter the associated callable object has (named *Input parameter*), and click on the Default Value button. Enter true as its value and confirm this entry by clicking on Set, as shown in **Figure 22**.

What's left is to set the jump target. For this, look at the result states: The values are equal, and the values are different. Now that you know that your comparison will always return equal, you can set the jump target for it. Let it simply point to the Approve Complaint step, as shown in **Figure 23**.

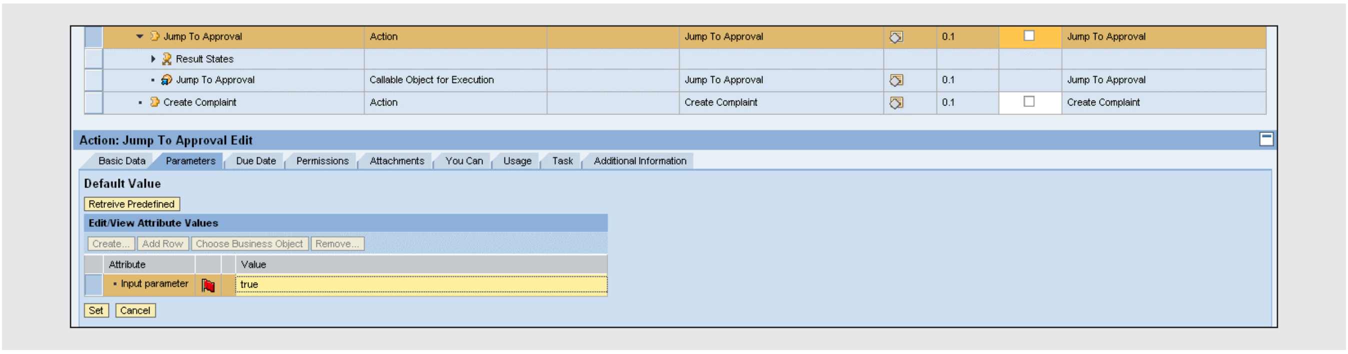
Perfect! This technique allows you to define any unconditional jump you need.


Create callable object

Comparison value: * true

Comparison value

Figure 21 Setting the comparison value



Jump To Approval	Action		Jump To Approval	0.1	☐	Jump To Approval
Result States						
Jump To Approval	Callable Object for Execution		Jump To Approval	0.1		Jump To Approval
Create Complaint	Action		Create Complaint	0.1	☐	Create Complaint

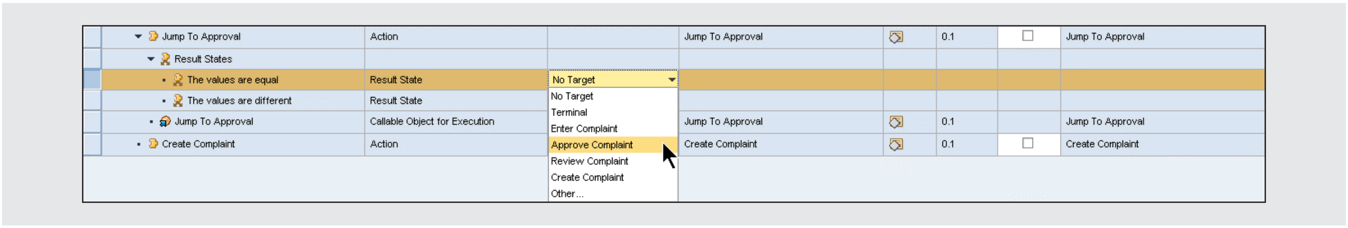
Action: Jump To Approval Edit

Default Value

Edit/View Attribute Values

Attribute	Value
Input parameter	true

Figure 22 Setting default values



Jump To Approval	Action		Jump To Approval	0.1	☐	Jump To Approval
Result States						
The values are equal	Result State	No Target				
The values are different	Result State					
Jump To Approval	Callable Object for Execution		Jump To Approval	0.1		Jump To Approval
Create Complaint	Action		Create Complaint	0.1	☐	Create Complaint

Figure 23 Unconditional jump back

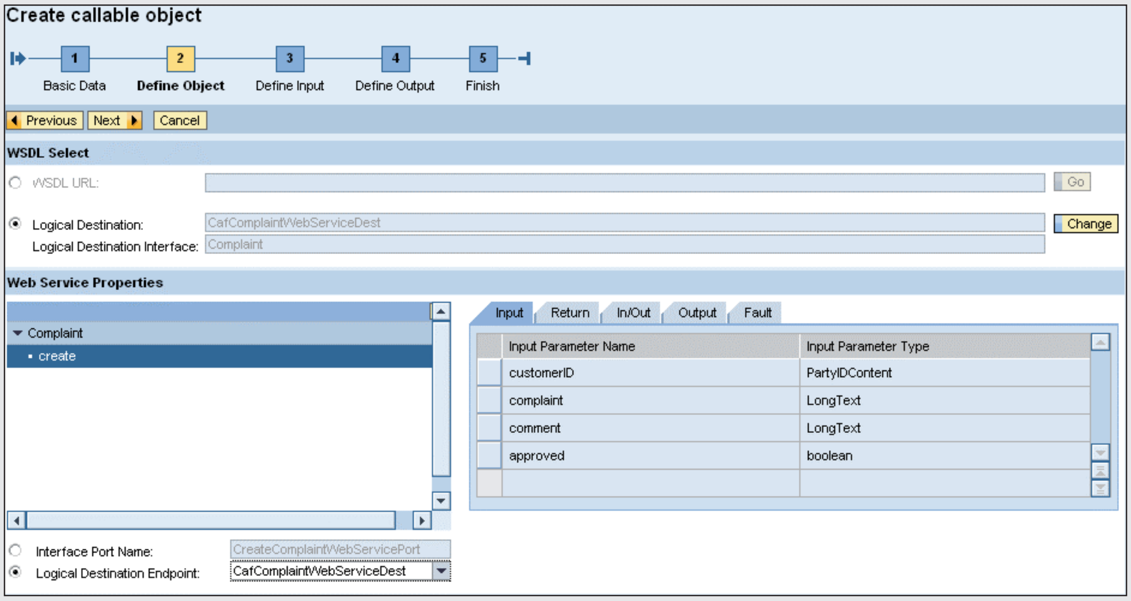
Now, let's continue the process-modeling exercise with the last step of the process: the service call. The idea is that once you are through with the complaint-handling between the call center agent and the complaint manager, you want to save the data in your database. This is an approach you should typically follow in your composite processes as well: Collect data in the SAP GP context, and once the data is in a stable state, save it for later reuse in the database.

Create Complaint

Since you want to assign the service call to the Create Complaint step, select the appropriate row in the process flow editor and open the New Callable Object wizard. This time, you select Web Service as the callable object type. It's located under the Service node (**Figure 20**). Provide the name Create Complaint and proceed to the next wizard step, as shown in **Figure 24**, where the logical destination comes into play: Activate the Logical Destination radio button and click on Change. (If you're calling the dialog for the first time, the button is labeled Select.)

A new input area appears containing a search form in which you can select the appropriate logical destination using the drop-down list from the In field. (This screen changes in its make-up quite often so you'll see the areas mentioned when you get to them.) Select the entry CafComplaintWebServiceDest and click on Search. A result table with available services shows the entry for your Complaint business object. Select the appropriate row of the result table and click on the Accept button. Next, you get a list of service operations for the business object. In this case, it only contains the Create method. Click on it and choose CafComplaintWebServiceDest from the drop-down list from Logical Destination Endpoint. By doing this, you have chosen the operation you want to assign to this step along with the endpoint location in which the service actually resides (**Figure 24**).

Now, you are ready to proceed: Click on Next. The rest is pretty straightforward. Just follow the wizard to its end, and don't forget to test your service on the Test tab of your callable object. Make sure that the service sends back reasonable return values; for example, see **Figure 25** on the next page.



WSDL URL:

Logical Destination: CafComplaintWebServiceDest

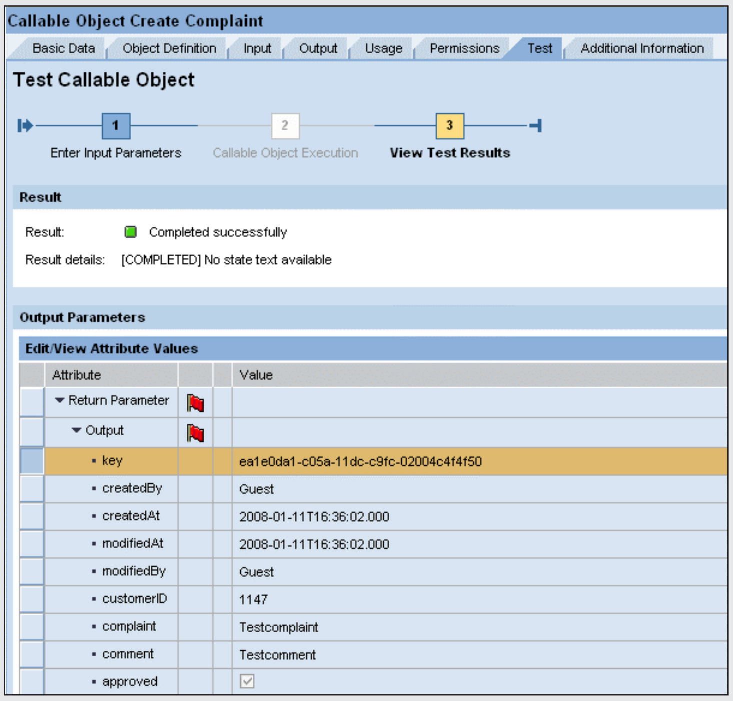
Logical Destination Interface: Complaint

Input Parameter Name	Input Parameter Type
customerID	PartyIDContent
complaint	LongText
comment	LongText
approved	boolean

Interface Port Name: CreateComplaintWebServicePort

Logical Destination Endpoint: CafComplaintWebServiceDest

Figure 24 Assigning a Web service to a callable object via logical destination



Test Callable Object

Result

Result: 🟩 Completed successfully

Result details: [COMPLETED] No state text available

Output Parameters

Edit/View Attribute Values

Attribute		Value
Return Parameter		
Output		
key		ea1e0da1-c05a-11dc-c9fc-02004c4f4f50
createdBy		Guest
createdAt		2008-01-11T16:36:02.000
modifiedAt		2008-01-11T16:36:02.000
modifiedBy		Guest
customerID		1147
complaint		Testcomplaint
comment		Testcomment
approved		✓

Figure 25 Testing the callable object for Web services

Additional configuration steps

All of the steps are now in place; you have defined the process flow including all jumps. What's left? Basically, two configuration steps that address the following questions are missing:

1. How do you make sure that the correct data is passed from step to step?

2. How do you assign roles to the steps so that the process framework knows whom to inform when a step is ready for execution?

Parameter consolidation

So far, you have associated callable objects to actions. Both input and output parameters have been associated with each of the steps, and these parameters together make up the SAP GP's context. Now, you need to tell SAP GPs how the parameter's values should flow from step to step. This is called *parameter consolidation* in SAP GP terms because you don't map each field from step to step. Consolidating parameters means treating all parameters assigned to each other as one parameter from the SAP GP's point of view. Let's see how this works in practice.

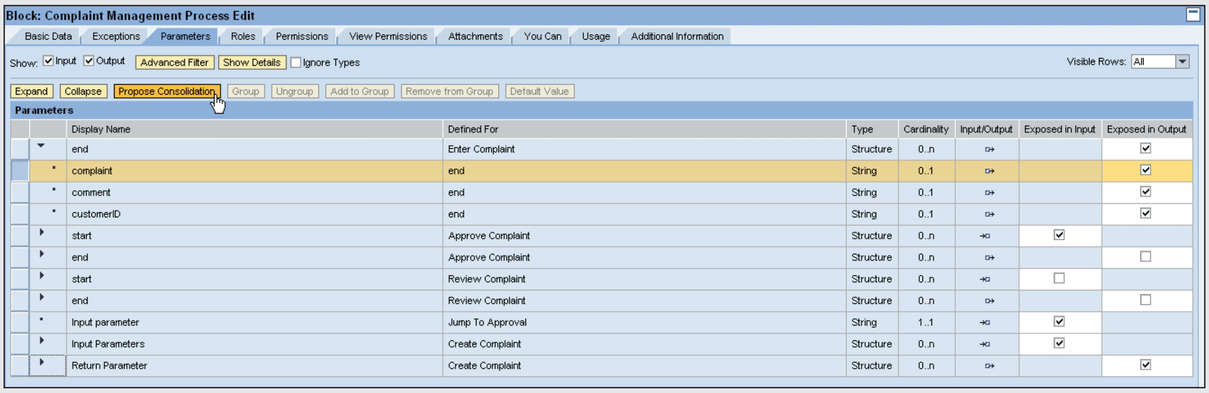
Select the block row of your process model in the process flow editor. Parameter consolidation only makes sense on the block level because the parameters of all steps are only visible to the block that contains them. Now, select the Parameters tab, which shows you all the parameters available for this particular block. In the Defined For column of the Parameters table, some entries appear twice while others don't because of the difference between input and output parameters. Callable objects with output (or input) parameters inevitably appear only once, whereas callable objects with input *and* output parameters appear twice. The Input/Output column tells you which parameters are input and which ones are output.

For example, expand the end node of the first table row by clicking on the triangle in front of it, as shown in **Figure 26**. The three output parameters of the Enter Complaint step appear. How should you use those parameters? Think about the process flow: The data entered in the first step should appear on the next screen for the complaint manager. Furthermore, the data should also appear on the call center agent's review screen, if needed. Finally, the data should be used to fill the interface of the Create Complaint Web service once you finish your process.

Therefore, you must make sure that the output parameters from the Enter Complaint step are consolidated to become the input parameters of all forthcoming steps. This defines the data flow inside the SAP GPs. You can do this manually by collecting all those fields individually (hold the Ctrl key down and select all the rows of those fields that you want to assign to each other), but this is rather cumbersome and error-prone. There is a better solution. Let's take the first field of the Enter Complaint step as an example:

1. Select the complaint field.
2. Click on the Propose Consolidation button (**Figure 26**).
3. Click on the Group button.
4. In the dialog that pops up, confirm the consolidation by clicking on Create.

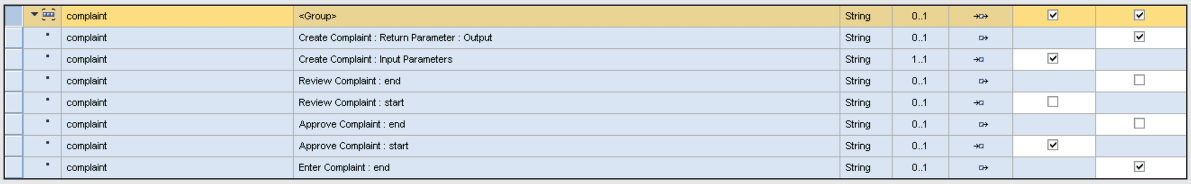
By doing this, the SAP GP framework automatically searches for all parameters with the same name and proposes them for consolidation. This trick saves you a lot of effort. Since you are in charge of defining all parameters for all steps of your process, you can heavily influence the effort needed to consolidate parameters. The trick is: If you use the same name for these parameters, they should be handled as one parameter across the process. This is a good reason to wrap external services in the services layer and provide names for parameters that SAP GP can easily consolidate later. Be aware of this every time you define your interfaces for screens and for Web services. As a result of the consolidation step, you get a new group that contains all collected fields (see **Figure 27**).



Show: ✓ Input ✓ Output ☐ Ignore Types — Visible Rows: All

Display Name	Defined For	Type	Cardinality	Input/Output	Exposed in Input	Exposed in Output
end	Enter Complaint	Structure	0..n	□→		✓
complaint	end	String	0..1	□→		✓
comment	end	String	0..1	□→		✓
customerID	end	String	0..1	□→		✓
start	Approve Complaint	Structure	0..n	→□	✓	
end	Approve Complaint	Structure	0..n	□→		☐
start	Review Complaint	Structure	0..n	→□	☐	
end	Review Complaint	Structure	0..n	□→		☐
Input parameter	Jump To Approval	String	1..1	→□	✓	
Input Parameters	Create Complaint	Structure	0..n	→□	✓	
Return Parameter	Create Complaint	Structure	0..n	□→		✓

Figure 26 Parameter consolidation in SAP GP



complaint	<Group>	String	0..1	→□→	✓	✓
complaint	Create Complaint : Return Parameter : Output	String	0..1	□→		✓
complaint	Create Complaint : Input Parameters	String	1..1	→□	✓	
complaint	Review Complaint : end	String	0..1	□→		☐
complaint	Review Complaint : start	String	0..1	→□	☐	
complaint	Approve Complaint : end	String	0..1	□→		☐
complaint	Approve Complaint : start	String	0..1	→□	✓	
complaint	Enter Complaint : end	String	0..1	□→		✓

Figure 27 Consolidated parameters

Repeat the parameter consolidation steps for the fields comment, customerID, and approved. As a final result, you should have the groups shown in **Figure 28**.

Role assignment

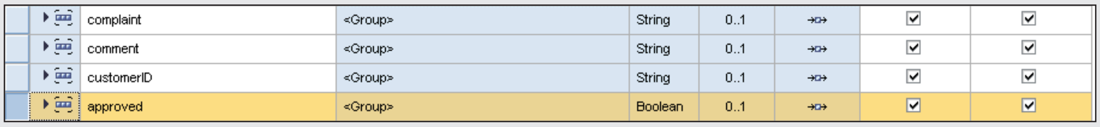
For each of the steps, you have to decide who should execute them. Due to your initial planning, you have the answers already: the call center agent should execute the steps Enter Complaint and Review Complaint, and the complaint manager should handle the Approve Complaint step. For background steps, you have to find out on whose behalf the service call should be executed. This is the kind of information for which you're looking. These are process roles, not the roles of users and groups defined in any of your user databases, Lightweight Directory Access Protocol (LDAP) systems, or other SAP systems.

Once you initiate a process, you assign concrete users or groups to the process roles, but during process-modeling you focus only on the process roles themselves. You define them on a block level by selecting the block row in the process flow editor and choosing the Roles tab, as shown in **Figure 29**. On that tab, a table that contains all of the available actions appears. You select those rows that the same process role executes and assign an appropriate name to the role. In the illustration, select Processor of Enter Complaint and Processor of Review Complaint rows (keep the Ctrl key pressed down while you select the rows), fill the Consolidate To input field with Call Center Agent, and click on Go.

For simplicity's sake, assign the remaining three rows to the Complaint Manager role.

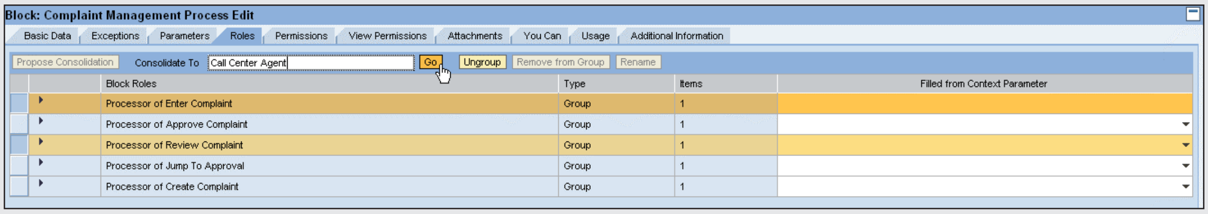
This ends the modeling of the process. Now, you only have to adjust some final parameters on the process level before you can kick off the process. Select the first row of the process flow editor, and then select the Parameters tab to display all process parameters. This time you don't want to consolidate parameters, you only want to make sure that the process can start without providing input parameters to it. In other words, processes can start and at the same time fill their own context with data. This is convenient if one process ends, kicks off another process, and passes parameters to it. However, you don't need to pre-fill in this example. That's why you remove all checkmarks in the Exposed in Input column.

Next, move to the Roles tab. Here, you get a list of all previously defined process roles extended by standard SAP GP roles. SAP GP roles are administrator, overseer, and owner. These roles have different rights during runtime (e.g., watching the state of a process or actively changing role assignments), and the role names are self-explanatory. However, you have to define who actually assumes these roles. If



complaint	<Group>	String	0..1	→⊙→	✓	✓
comment	<Group>	String	0..1	→⊙→	✓	✓
customerID	<Group>	String	0..1	→⊙→	✓	✓
approved	<Group>	Boolean	0..1	→⊙→	✓	✓

Figure 28 Finished parameter consolidation



Block: Complaint Management Process Edit

Consolidate To: Call Center Agent

Block Roles	Type	Items	Filled from Context Parameter
Processor of Enter Complaint	Group	1	
Processor of Approve Complaint	Group	1	
Processor of Review Complaint	Group	1	
Processor of Jump To Approval	Group	1	
Processor of Create Complaint	Group	1	

Figure 29 Consolidating roles

you click on the associated Role Type drop-down list, you have two or three choices:

- **Initiator:** The one who kicks off the process should take over this role as well.
- **Initiation defined:** Once a process is kicked off, the assignment to a concrete user or role has to be done explicitly.
- **Runtime defined:** The assignment will be determined during runtime by some calculation (this option is available for process roles only).

In this case, you assign Initiator to all roles but the complaint manager, who should have the role type Initiation Defined, as shown in **Figure 30**.

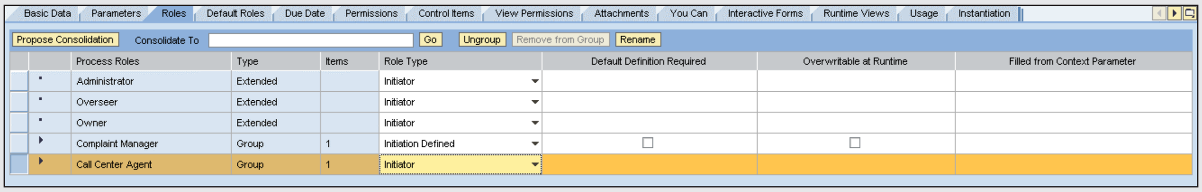
The last tab you need to adjust before you start your process is the Default Roles tab. For every role that has the assignment Initiation Defined, you can assign users or groups that the process should use when no users or groups are assigned to it during process start. In this case, you can assign any user from your user store to this role. The only prerequisite is that the user must have the rights to execute SAP GPs. You just click on the Add Default button, search your user database for an appropriate user, and click on the Add button to assign him or her to the complaint manager process role, as shown in **Figure 31**.

Note!

If you don't want to switch between different roles and see the notifications being sent among the process participants, you can set the role type of the Complaint Manager to Initiator.

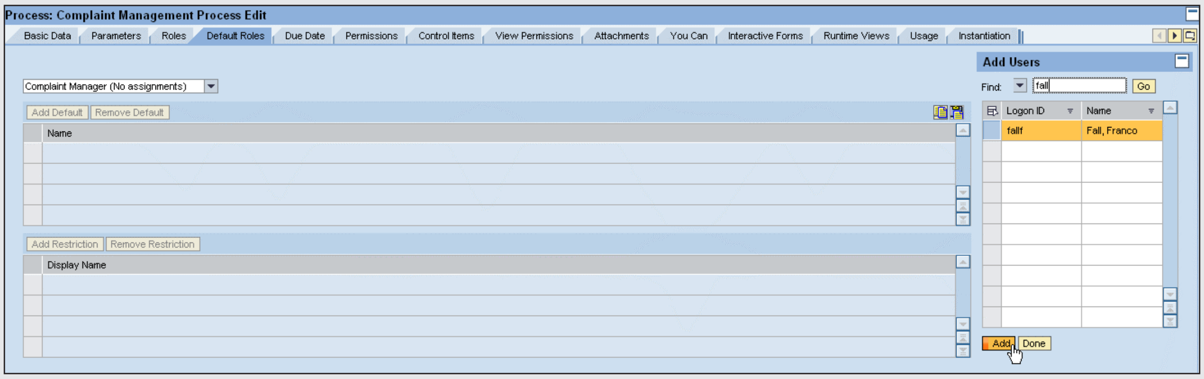
Executing the process

Before you start your process, activate it by clicking on the Activate button () in the upper toolbar. Now



Process Roles	Type	Items	Role Type	Default Definition Required	Overwritable at Runtime	Filled from Context Parameter
Administrator	Extended		Initiator			
Overseer	Extended		Initiator			
Owner	Extended		Initiator			
Complaint Manager	Group	1	Initiation Defined	☐	☐	
Call Center Agent	Group	1	Initiator			

Figure 30 Role assignment on the process level



Logon ID	Name
fallf	Fall, Franco

Figure 31 Adding a default role

you're ready to initiate the process. Select the Instantiation tab of the process, check the checkboxes for Include Default Parameters and Start Process Automatically, click on the Generate Instantiate URL button, and click on Open Instantiate Application, as shown in **Figure 32**.

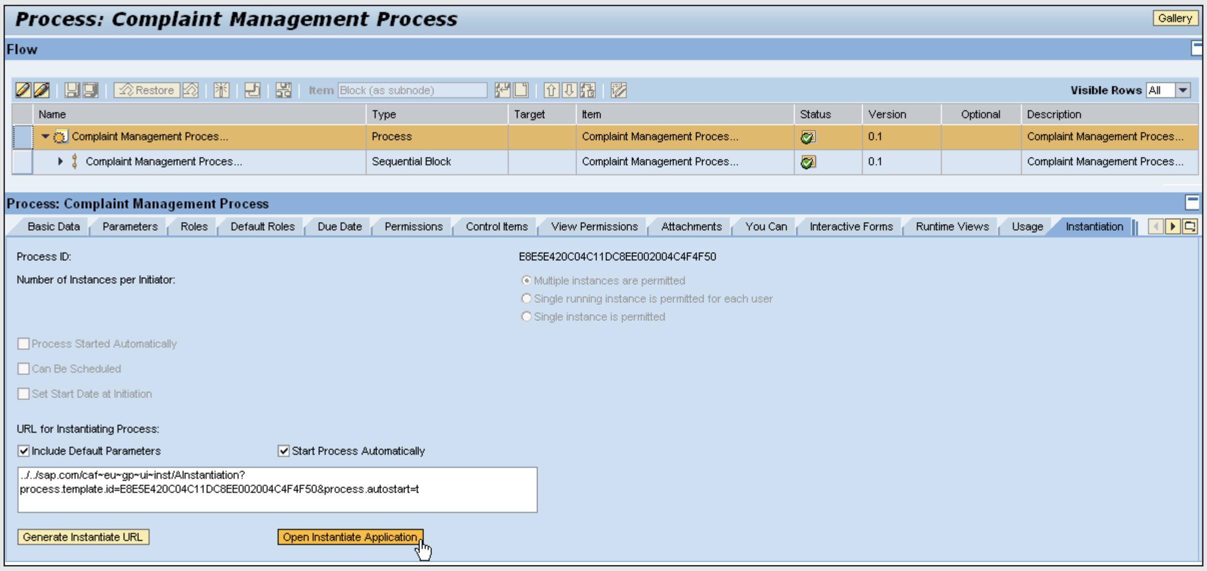
By providing these settings, you ensure the immediate start of the process without opening a dialog upfront to ask for roles or parameters you can set manually. You should also recognize the URL being created for you. You can use it within other Web pages to start a process from there.

Once the process has been started, you are automatically guided to the Enter Complaint screen as you defined the process initiator to also be the processor of the first step. The screen modeled in SAP NetWeaver Visual Composer appears, encapsulated by a standard SAP GP frame, as shown in **Figure 33**.

Fill in some sample data and click on Send to Complaint Manager. This action passes the data to the SAP GP's framework, which is responsible for sending out a notification to the next process participant. If you've set the role type of the complaint manager role to Initiated, you are automatically guided to the next step. Otherwise, you have to log on to the portal as the user you chose during default role assignment. Once logged on, choose Guided Procedures from the first-level navigation and Runtime from the second. You reach the Guided Procedures Runtime Work Center, as shown in **Figure 34** on page 64, from which you navigate to the next step.

Just follow the link to the tasks that require your action and from there navigate to the Approval screen. Verify that the data entered during step 1 appears correctly on the Complaint Manager's screen. This ensures that parameter consolidation works correctly.

From there, it's up to you whether you want to hand back the request to the call center or immediately end the process by clicking on Finished.



	Name	Type	Target	Item	Status	Version	Optional	Description
	Complaint Management Proces...	Process		Complaint Management Proces...	✓	0.1		Complaint Management Proces...
	Complaint Management Proces...	Sequential Block		Complaint Management Proces...	✓	0.1		Complaint Management Proces...

Process ID: E8E5E420C04C11DC8EE002004C4F4F50

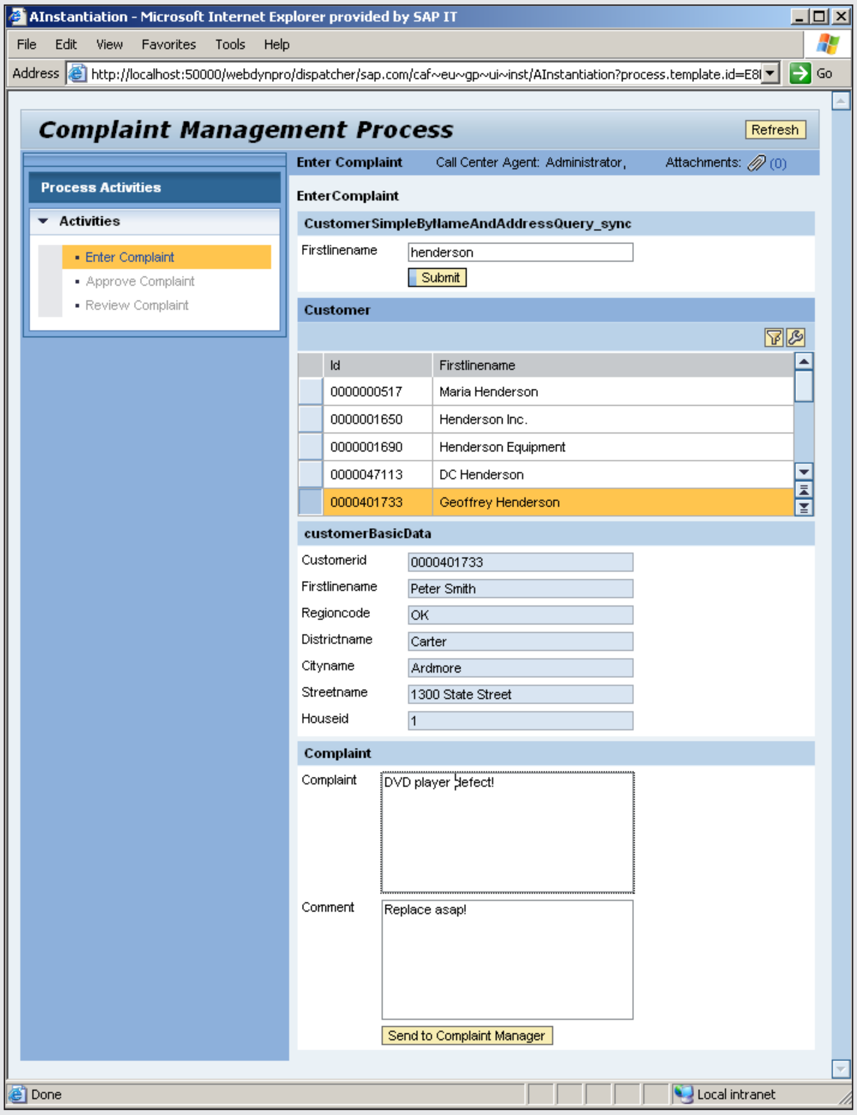
Number of Instances per Initiator: [✓ Multiple instances are permitted] Single running instance is permitted for each user / Single instance is permitted

☐ Process Started Automatically
☐ Can Be Scheduled
☐ Set Start Date at Initiation

URL for Instantiating Process:
☑ Include Default Parameters ☑ Start Process Automatically

```
../../sap.com/caf~eu~gp~ui~inst/AInstantiation?
process.template.id=E8E5E420C04C11DC8EE002004C4F4F50&process.autostart=t
```

Figure 32 Initiating the complaint management process



http://localhost:50000/webdynpro/dispatcher/sap.com/caf~eu~gp~ui~inst/AInstantiation?process.template.id=E8I

Complaint Management Process

Enter Complaint   Call Center Agent: Administrator,

EnterComplaint

CustomerSimpleByNameAndAddressQuery_sync

Firstlinename	henderson

Customer

Id	Firstlinename
0000000517	Maria Henderson
0000001650	Henderson Inc.
0000001690	Henderson Equipment
0000047113	DC Henderson
0000401733	Geoffrey Henderson

customerBasicData

Customerid	0000401733
Firstlinename	Peter Smith
Regioncode	OK
Districtname	Carter
Cityname	Ardmore
Streetname	1300 State Street
Houseid	1

Complaint

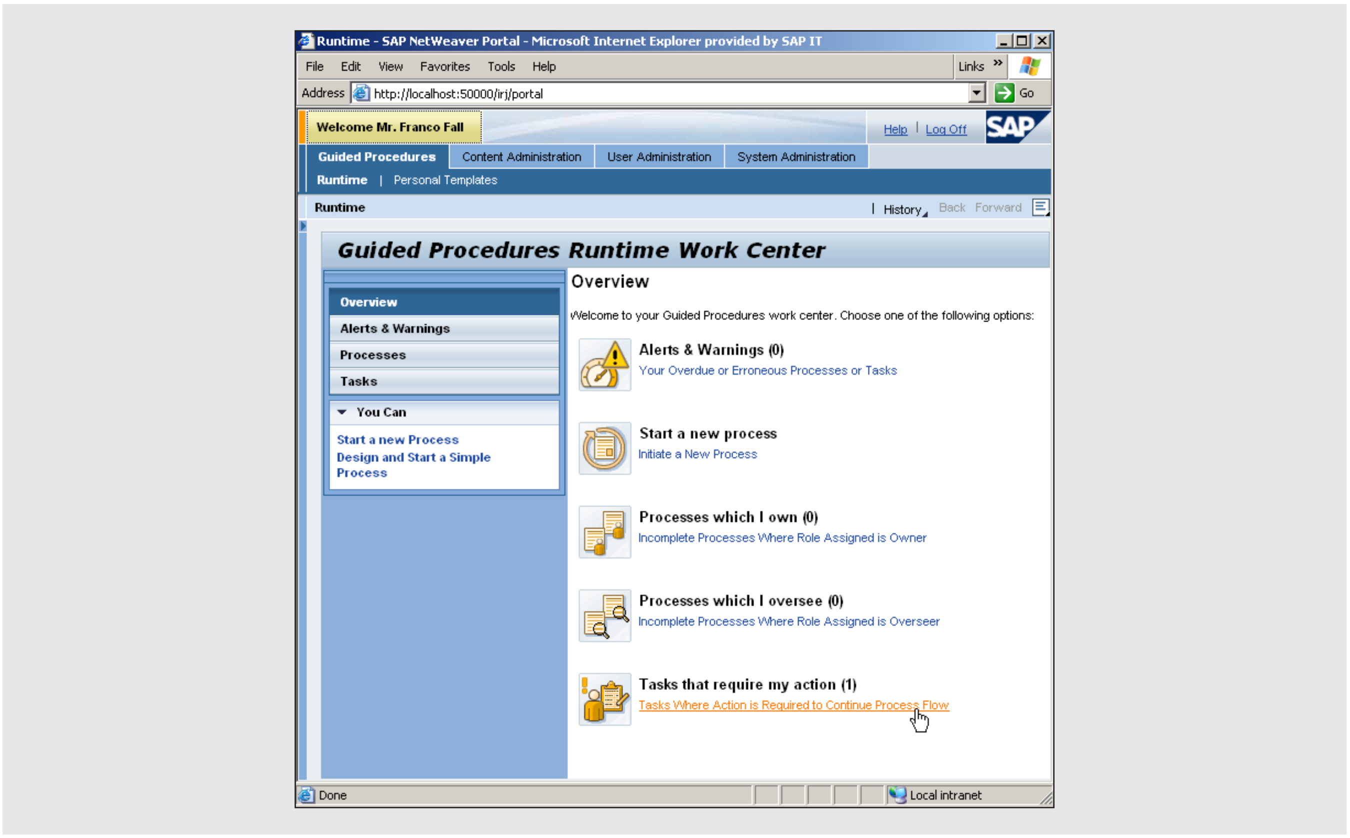
Complaint	DVD player defect!
Comment	Replace asap!

Figure 33 Enter Complaint step during runtime

However, once you finish your process, make sure that it added an appropriate entry to your database; to do this, you use the findAll method of the Complaint business object (**Figure 10**). With this knowledge, you are now well-equipped to start your own journey into the world of composite applications.

Conclusion

Congratulations! This has been a long and interesting journey across the world of composite applications and their development. I gave you the concept of composite applications, their characteristics, and


Runtime - SAP NetWeaver Portal - Microsoft Internet Explorer provided by SAP IT

File Edit View Favorites Tools Help

Address http://localhost:50000/irj/portal

Welcome Mr. Franco Fall

Help | Log Off

Guided Procedures | Content Administration | User Administration | System Administration

Runtime | Personal Templates

Runtime

History Back Forward

Guided Procedures Runtime Work Center

Overview

Alerts & Warnings

Processes

Tasks

You Can

Start a new Process

Design and Start a Simple Process

Overview

Welcome to your Guided Procedures work center. Choose one of the following options:

Alerts & Warnings (0)
Your Overdue or Erroneous Processes or Tasks

Start a new process
Initiate a New Process

Processes which I own (0)
Incomplete Processes Where Role Assigned is Owner

Processes which I oversee (0)
Incomplete Processes Where Role Assigned is Overseer

Tasks that require my action (1)
Tasks Where Action is Required to Continue Process Flow

Done

Local intranet

Figure 34 The Guided Procedures Runtime Work Center

their architecture combined with a methodology about how to approach them from a development perspective. In addition, I have included best practices and recommendations throughout the text to support you in your own development efforts. I hope I whetted your appetite for more information about composite applications and their development. In Part 1 of the series, I used a dummy service to implement the functionality to Read Customer Data and promised to explain how you can replace this dummy service with a real enterprise service. I will cover this aspect of composite application development in more detail in a forthcoming issue of *SAP Professional Journal*; this technique unveils the real power of loosely coupled applications.

Composite applications play an important role in SAP's enterprise SOA strategy, and SAP NetWeaver CE is becoming a solid player in SAP's product portfolio. Expect major enhancements and improvements for SAP NetWeaver CE in the years to come to ease the developer's life even further. If you found this article useful and you are interested in more details, let me know your thoughts and areas of interest. Watch for future installments of this publication in which I'll delve more deeply into other composite application topics.

The enterprise SOA idea changes the way we will develop software in the future. With SAP NetWeaver CE, SAP provides an environment that supports you in your efforts to implement innovative collaborative processes on top of existing IT landscapes. The importance of composite applications will only increase over time, and you should become a part of this successful story.