Oracle® Banking Platform

SOA Extensibility Guide Release 2.7.1.0.0 F16199-01

March 2019



Oracle Banking Platform SOA Extensibility Guide, Release 2.7.1.0.0

F16199-01

Copyright © 2011, 2019, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate failsafe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

This software or hardware and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

| Preface | 8 |
|--|----|
| Audience | 8 |
| Documentation Accessibility | 8 |
| Related Documents | 8 |
| Conventions | 8 |
| 1 Objective and Scope | 11 |
| 1.1 Overview | 11 |
| 1.2 Objective and Scope | 11 |
| 1.2.1 Extensibility Objective | 11 |
| 1.3 Complementary Artefacts | 11 |
| 1.4 Out of Scope | 12 |
| 2 Overview of Use Cases | 13 |
| 2.1 Extensibility Use Cases | 13 |
| 2.1.1 SOA Customization | 13 |
| 3 Workflow Task Routing and Assignment Rules | 15 |
| 3.1 Workflow Task Routing and Assignment Rules | 15 |
| 3.2 Origination Human Tasks | 15 |
| 3.3 Origination Approval Human Tasks | 15 |
| 4 Workflow Analytics | 19 |
| 4.1 Architecture | 19 |
| 4.2 OBP Workflow Analytics Extension Points | 19 |
| 4.2.1 Introduction of new human task | 19 |
| 4.2.2 Addition of custom business indicator(s) to existing data object | 20 |
| 4.2.3 Publishing analytics data to system other than Oracle BAM | 20 |

| | 4.2.4 Publishing new DataObject from OBP Domain | 20 |
|---|---|------|
| 5 | SOA Customizations | .23 |
| | 5.1 Customization Layer | .23 |
| | 5.2 Customization Class | .24 |
| | 5.3 Enabling Application for Seeded Customization | .25 |
| | 5.4 SOA Customization Example Use Cases | . 26 |
| | 5.4.1 Add a Partner Link to an Existing Process | .26 |
| | 5.4.2 Add a Human Task to an Existing Process | .44 |

List of Figures

| Figure 2–1 SOA Customization | 13 |
|---|----|
| Figure 3–1 SM500 page | 16 |
| Figure 3–2 SM500 page with appropriate Datatype | 16 |
| Figure 3–3 To Fetch Attribute Value | 17 |
| Figure 3–4 Custom Attributes | 18 |
| Figure 3–5 Redeploy process | |
| Figure 4–1 Architecture Diagram | 19 |
| Figure 5–1 Add an entry for new Customization Layer | 24 |
| Figure 5–2 Create Customization Class | 25 |
| Figure 5–3 Enabling Application for Seeded Customization | 26 |
| Figure 5–4 Select SOA Project | |
| Figure 5–5 Enter SOA Project Name | |
| Figure 5–6 Configure SOA Settings | |
| Figure 5–7 Create Mediator | |
| Figure 5–8 Select Target Type | |
| Figure 5–9 Request Transformation Map to create new mapper file | 31 |
| Figure 5–10 Mapping Input and Output string | 31 |
| Figure 5–11 Select Deployment Action | |
| Figure 5–12 Deploy Configuration Settings | |
| Figure 5–13 Select Deployment Server | |
| Figure 5–14 Select Target SOA Server | |
| Figure 5–15 Select SOA Domain | 35 |
| Figure 5–16 Test Web Service | |
| Figure 5–17 Customization of SOA Application - Flow | |

| Figure 5–18 Customization of SOA Application - Notify Customer | |
|--|----|
| Figure 5–19 Add Partner Link Component | |
| Figure 5–20 Add Invoke Component | 40 |
| Figure 5–21 Edit Copy Rules Variable | 41 |
| Figure 5–22 Add Assign Components - Reply | 42 |
| Figure 5–23 Design View of the BPEL Process | 42 |
| Figure 5–24 Test Customized Composite - Flow | |
| Figure 5–25 Test Customized Composite - invokeEchoService | 44 |
| Figure 5–26 Select SOA Project | 45 |
| Figure 5–27 Create SOA Project Name | 45 |
| Figure 5–28 Configure SOA Settings | 46 |
| Figure 5–29 Configure BPEL Process Settings | 46 |
| Figure 5–30 Enter Human Task Name | 47 |
| Figure 5–31 Create Human Task - General Tab | 47 |
| Figure 5–32 Add Human Task Parameter | 48 |
| Figure 5–33 Create Human Task - Data Tab | 48 |
| Figure 5–34 Add Participant Type Details | |
| Figure 5–35 Create Human Task - Assignment Tab | 49 |
| Figure 5–36 Select Human Task Parameters | 50 |
| Figure 5–37 Create Human Task - Delete Condition | 50 |
| Figure 5–38 Create Human Task - Expression Builder | 51 |
| Figure 5–39 Create Human Task - Copy Rules | 51 |
| Figure 5–40 Create Human Task - BPEL Process | 52 |
| Figure 5–41 Select Human Task Form | 53 |
| Figure 5–42 Select Human Task Form Deployment Action | 54 |
| Figure 5–43 Select Human Task Form - Weblogic Options | 54 |

| Figure 5–44 Add Customization Scope to SOA Application | 55 |
|---|----|
| Figure 5–45 Add Partner Link Component | 56 |
| Figure 5–46 Add Invoke Component | 57 |
| Figure 5–47 Add Receive Component using BPEL functions | 58 |
| Figure 5–48 Add Assign Component | 59 |
| Figure 5–49 Deploy and Test Customized SOA Composite - My Tasks Tab | 60 |
| Figure 5–50 Deploy and Test Customized SOA Composite - Flow | 60 |
| Figure 5–51 Deploy and Test Customized SOA Composite - Invoke Input | 61 |
| Figure 5–52 Deploy and Test Customized SOA Composite - Receive Output | 61 |

Preface

This guide explains customization and extension of Oracle Banking Platform.

This preface contains the following topics:

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

Audience

This guide is intended for the users of Oracle Banking Platform.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/us/corporate/accessibility/index.html.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/us/corporate/accessibility/support/index.html#info or visit http://www.oracle.com/us/corporate/accessibility/support/index.html#trs if you are hearing impaired.

Related Documents

For more information, see the following documentation:

- For installation and configuration information, see the Oracle Banking Platform Localization Installation Guide - Silent Installation guide.
- For a comprehensive overview of security, see the Oracle Banking Platform Security Guide.
- For the complete list of licensed products and the third-party licenses included with the license, see the Oracle Banking Platform Licensing Guide.
- For information related to setting up a bank or a branch, and other operational and administrative functions, see the Oracle Banking Platform Administrator Guide.
- For information on the functionality and features, see the respective Oracle Banking Platform Functional Overview documents.
- For recommendations of secure usage of extensible components, see the Oracle Banking Platform Secure Development Guide.

Conventions

The following text conventions are used in this document:

| Convention | Meaning |
|---|--|
| boldface Boldface type indicates graphical user interface elements associated with action, or terms defined in text or the glossary. | |
| italic | Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values. |
| monospace | Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter. |

1 Objective and Scope

This chapter defines the objective and scope of this document.

1.1 Overview

Oracle Banking Platform (OBP) is designed to help banks respond strategically to today's business challenges, while also transforming their business models and processes to reduce operating costs and improve productivity across both front and back offices. It is a one-stop solution for a bank that seeks to leverage Oracle Fusion experience for its core banking operations, across its retail and corporate offerings.

OBP provides a unified yet scalable IT solution for a bank to manage its data and end-to-end business operations with an enriched user experience. It comprises pre-integrated enterprise applications leveraging and relying on the underlying Oracle Technology Stack to help reduce in-house integration and testing efforts.

1.2 Objective and Scope

Most product development can be accomplished through highly flexible system parameters and business rules. Further competitive differentiation can be achieved through IT configuration and extension support. In OBP, additional business logic required for certain services is not always a part of the core product functionality but could be a client requirement. For these purposes, extension points and customization support have been provided in the application code which can be implemented by the bank and / or by partners, wherein the existing business logic can be added with or overridden by customized business logic. This way the time consuming activity of custom coding to enable region specific, site specific or bank specific customizations can be minimized.

1.2.1 Extensibility Objective

The broad guiding principles with respect to providing extensibility in OBP are summarized below:

- Strategic intent for enabling customers and partners to extend the application.
- Internal development uses the same principles for client specific customizations.
- Localization packs
- Extensions by Oracle Consultants, Oracle Partners, Banks or Bank Partners.
- Extensions through the addition of new functionality or modification of existing functionality.
- Planned focus on this area of the application. Hence, separate budgets specifically for this.
- Standards based OBP leverages standard tools and technology
- Leverage large development pool for standards based technology.
- Developer tool sets provided as part of JDeveloper and Eclipse for productivity.

1.3 Complementary Artefacts

The document is a developer's extensibility guide and does not intend to work as a replacement of the functional or technical specification, which would be the primary resource covering the following:

- OBP Zen training course
- OBP installation and configuration
- OBP parameterization as part of implementation
- Functional solution and product user guide

References to plugin indicate the eclipse based OBP development plugin for relevant version of OBP being extended. The plugin is not a product GA artefact and is a means to assist development. Hence, the same is not covered under product support.

1.4 Out of Scope

The scope of extensibility does not intend to suggest that OBP is forward compatible.

2 Overview of Use Cases

The use cases that are covered in this document shall enable the developer in applying the discipline of extensibility to OBP. While the overall support for customizations is complete in most respects, the same is not a replacement for implementing a disciplined, thoughtful and well-designed approach towards implementing extensions and customizations to the product.

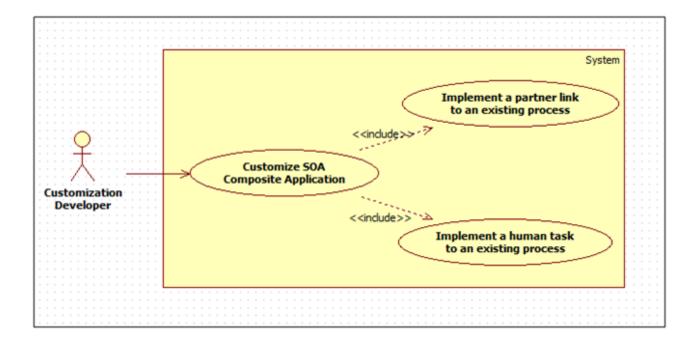
2.1 Extensibility Use Cases

This section gives an overview of the extensibility topics and customization use cases to be covered in this document. Each of these topics is detailed in the further sections.

2.1.1 SOA Customization

OBP Application provides the feature for customizing SOA composite applications based on the additional requirements which may vary from client to client. It includes implementing the partner link to an existing process or implementing human tasks or sub processes which can be hooked into an existing product process.

Figure 2–1 SOA Customization



3 Workflow Task Routing and Assignment Rules

This chapter covers the extensibility aspects of business rules used in SOA.

3.1 Workflow Task Routing and Assignment Rules

As part of overall origination workflow, different human tasks are created. Each human task can be routed to specific assignees. Such task assignment can be configured using a business rules. All such business rules are logically grouped together into different BPEL composites based on the functionality. Specific list of facts are provided out of the box for rule configuration. However, more facts can be made available for rule configuration by customizing specific application services which return the list of facts.

Below sections provide the information about services which can be extended to include more custom facts.

3.2 Origination Human Tasks

Custom facts can be added to the response of below application service using post extensions:

com.ofss.fc.appx.origination.service.core.OriginationBusinessIndicatorApplicationServiceSpi

Key names used to populate the dictionary object of the service response are made available automatically as custom facts for configuration of business rule. These custom attributes can be accessed in the rule configuration as below:

getCustomTextAttribute(customAttributes,"<Custom Attribute Name>")

3.3 Origination Approval Human Tasks

Approval workflow routing rules can be extended to include existing or entirely new attributes.

This is demonstrated using an example:

For structure solution approvals, routing to be setup based on whether the task includes a secured product.

Use case:

Table 3–1 Use Case

| Servic e-id | com.ofss.fc.appx.origination.service.lending.core.application.LendingApplicationServiceSpi.confirmStructureSolution |
|-----------------------------|---|
| Appro val Proces s | com.ofss.fc.approval.lending_confirmstructuresolution |
| Custo m attribut e | LendingStructureSolution_IsTaskIncludesSecuredProduct |

Broad steps to use custom attributes in routing rule:

- Step 1: Create custom attribute.
- Step 2: Configure adapter name that will be used to retrieve value of the custom attribute.
- Step 3: Develop the adapter.
- Step 4: Modify routing rule to include the custom adapter.

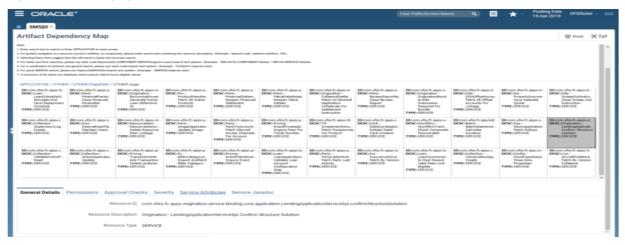
Step 1: Create the new attribute using Artifact Dependency Map (Fast Path: SM500) page

This configuration ensures that OBP fetches the name for the custom attribute as part of its attribute evaluation framework when confirm-structure-solution is executed.

Step 1.A: Search for below service ID in Artifact Dependency Map (Fast Path: SM500) page

'com.ofss.fc.appx.origination.service.lending.core.application.LendingApplicationServiceSpi.confirmS tructureSolution'

Figure 3–1 SM500 page



Step 1.B: Add attribute name 'LendingStructureSolution_IsTaskIncludesSecuredProduct'

With appropriate datatype in 'AllowedPolicyAttributes' under 'Service Attributes' tab in SM500 page as shown below:

Figure 3–2 SM500 page with appropriate Datatype

| tifact Deper | ndency Map | | | | | | | | | | @ Print 3 |
|---|--|---|---|--|---|--|--|--|--|---|--|
| LoanScheduleSi Simulate And Save Repayment Schedule PE: SERVICE | PreancialParam Close Financial Parameter TYPPE: SERVICE | BernsuingPower Generate Home Loan Reference M TYPE: SERVICE | Product/Tenafac Fetch All Active Products TYPPE: SERVICE | Prancia/Datem Respen Financial Statement TYPPE/SERVICE | PatraDetailsApp Raspen Patra Details TYPPE:SERVICE | CaleteralSettle Petch Un Broked Application Caleterals For Settlement Instruction | BureauReportin Close Bureau Report TYPPE:SERVICE | OriginationBand Is Edit Submission Required Por Bandle TYPE:SERVICE | DDAOffsetAccos Fetch Al Offset Accounts For Unkage TYPE: SERVICE | PropertySneuran Save Selected Quote TYPE:SERVICE | SweepOutBrutru Clase Sweep Out Jadhuction TYPPE: SERVICE |
| topen, effss fic. epper, c Stochalerstein – Supervisiony Leg Create PE: SERVICE | IDroom.ofgs.fc.apgs.s. DESC:5ms - Deternationeffa. Handain Users TYPE:SERVICE | EDicom.ofts.fc.appi.re. DESC:desourceitem - Resourceitem: Delete Resource Delete Resource Delete Resource Delete Resource TYPE:SERVICE | IDocom, efis.fc.apping DESC/Party - Image/opticatio Update Image TYPE: SERVICE | IDicom, efss.fc.appix.g., DESC/Party - Party ToAccount Fetch Allowed Access Channels Par Account TYPE: SERVICE | IDicon of the fic apps p DESC/Pricing - EventPrical ppli- Inquire Fees For Ticket Number TYPE: SERVICE | IDicon.ofss.fc.apps.hd. DESC/14 - SeventmentSimu Fetch.Finguencies For Product TYPE:SERVICE | IDicom.efs.fc.appix.c. DESC:Card - DebtCardApplic. Initiate Debit Card Creation TYPE: SERVICE | IDicon of is fc appr.m. DESC/IIIvidea - Worldow Track. Check Compastie Recoverable TYPE: SERVICE | IDicen, ofs.fc.app.bat. DESC:Batch - BatchGSabatcok. Calculate Duration TYPE: SERVICE | ID:com.efst.fc.appx.e DESC:0ps - NounApplication Fetch Rollaps TYPE:SERVICE | ID.com.efs.fc.apps.or. DESC:Organition - Landrophysicati Confirm Structure Solution TYPE: SERVICE |
| ozem.ofia.5c.appi.c. SCCollection - Volidate/JosterP Read PE:SERVICE | ID-com.efis.fc.apps.c. DESC:Collection - ActionApplicatio. Update TYPE:SERVICE | IDroser.ofis.fc.apps.pr DESC/Pricing - TransactionOet. Add Transaction Octails Analysis TYPE:SERVICE | IDrown oftic fic approp. DESC/Fic - Biller CategoryA., Impact Andfatch Biller Category TYPE: SERVICE | ID-com, effor fic apps.gr DESC/Promg - EventPlaintenan, Inquire Event TYPE: SERVICE | IDrosen oftis Foappelie DESCILION - LoomApplication. Validate Lean Account Configuration 30m TYPE: SERVICE | IDicon offs.fc.apps.p. DESC/Party - PartyLastLatit. Patch Party Last Adulty TYPE: SERVICE | ID-com.effis.fc.appx.in. DESC(Ins - DescanceSimpl Patch By Vanian TYPE:SERVICE | IDrosm ofus fic apps.lo DESCILION - LoamCommonAc Is Must Resent Valid Rate Lock Bigble TYPE: SERVICE | IDroam offis Roapproc DESC:Callection - Sendonikoonig. Onada TYPPE:SERVICE | IDroom.efis.fc.app.co. DESC:Corfig - Drafhoperteak. Class Dros Properties TYPE:SERVICE | IDrozwi, ofisi, fic.appro.3c. DESCS.cm - AircraftCollators, Patch by Version Collatoral TYPE::SERVICE |
| | | roval Checks Sev | erity Service Attr | ibutes Service Ja | wadoc | | | | | | |
| | K 🖂 🖃 Detacl | h Value | | | | | | | | | |
| | | | | | | | | | | | |
| Name | yAttributes | LendingMatrix_VarRa | stehtakhangin,Lendin | gMatrix_LTVCategory | Groups,LendingMatri | x_PeeModification typ | e_LendingMatrix_Mar | netEntityPricing_IsPe | ervegotiated,Lending | Matrix_PixedHateHea | son,LendingMatrix |
| AllowedPolic | | | | | | | | | | | |
| | | | | | | | | | | | |
| AllowedPolic + × LendingStructure | | | | | | | | | | | |
| AllowedPollo AllowedPollo + × LendingStructureS LendingStructureS | Solution, booles | iecuredProduct | | | | | | | | | |

Step 2: Define the attribute in Constraint Attribute config with associated adapter to fetch attribute value

This configuration step ensures that the attribute retrieval framework executes the appropriate adapter to fetch the value of the custom attribute.

In flx_fw_config_all_b, insert the mapping of attribute and it's associated adapter.

Example:

Use the below insert statement to map attribute and the associated adapter developed to supply attribute value.

Figure 3–3 To Fetch Attribute Value

Insert into FLX_FW_CONFIG_ALL_B (PROP_ID,CATEGORY_ID,PROP_VALUE,FACTORY_SHIPPED_FLAG,PROP_COMMENTS,SUMM ARY_TEXT,CREATED_BY,CREATION_DATE,LAST_UPDATED_BY,LAST_UPDATED_DATE,OBJE CT_STATUS_FLAG,OBJECT_VERSION_NUMBER) values ('LendingStructureSolution_IsTaskIncludesSecuredProduct','ConstraintAttributeHelper','<Customadapter-name-as-per-extension-naming-standards>','N','<description>','<create-user>',SYSDATE, <create-user>'),'A',1);

The category_id 'ConstraintAttributeHelper' (or an appropriate override) in Preferences.xml will be used to retrieve the adapter added.

Example:

<Preference name="ConstraintAttributeHelper" PreferencesProvider="com.ofss.fc.infra.config.impl.DBBasedPropertyProvider"

parent="jdbcpreference"

propertyFileName="select prop_id, prop_value from flx_fw_config_all_b where category_id = 'ConstraintAttributeHelper'" syncTimeInterval="1800000" />

Step 3: Add derivation logic to set the attribute value in Adapter that is mapped in config

Develop the adapter mentioned in Step 2, method name to be named appropriately.

Example:

If adapter-name in Step 2 = 'FacilityApprovalDataMatrixAdapter' the method name will be

'getIsTaskIncludesSecuredProduct'

Step 4: Using custom fact in business rule

- a. Extract the rules process corresponding to the approvals process:OriginationAndFulfillment\process\com.ofss.fc.approval.genericrulesapprovalspi.executeapp rovalrulesoriglen\SOA\oracle\rules\HT_LendingSpi_ConfirmStructureSolutionRules.rules
- b. Open .rules file (HT_LendingSpi_ConfirmStructureSolutionRules.rules).
- c. In ruleset, modify rule with the desired condition using customAttributes (see example and screenshot below)

Figure 3–4 Custom Attributes

customAttributes is a CustomAttributes and getCustomTextAttribute(customAttributes,"LendingStructureSolution_IsTaskIncludesSecuredP roduct")=="Y"

d. Redeploy process

Figure 3–5 Redeploy process

| | n, approximation and approximate consistent provide a specific constraint and a specific const | <u> </u> |
|--|---|----------|
| Eie Edit View Application Relator Search Nevigate Build Bun Team | | |
| | Qpi facto | |
| Applications | | |
| Clightstonks/Fulliment * * | | 3 |
| Image: Image: Imag | Constraints Constrain | |
| | Low Issues: NT_LandingEpt_Confert@suctureEndelionRules.nates - Issues | |
| | | |
| ≥ Darks Constraints | | |
| A Recent Files | No bears | |
| RT_LendingSpi_ConfirmStructureSolutionRules rules - Structure Calls | | |
| Its Structure | | |

4 Workflow Analytics

This chapter explains the workflow analytics of Oracle Banking Platform (OBP).

4.1 Architecture

The following architecture diagram depicts the instrumentation approach for BPEL processes as well as human workflows. It also illustrates the approach to collect business indicators and the mechanism to publish them to the analytics system.

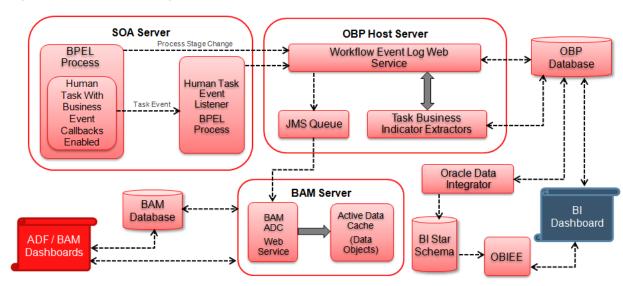


Figure 4–1 Architecture Diagram

4.2 OBP Workflow Analytics Extension Points

This section explains the extension points for OBP workflow analytics.

4.2.1 Introduction of new human task

The following steps can be performed for introducing new human task.

- 1. Enable business events in .task file.
- 2. Attribute "task Key" to be added in task payload. This is hash (#) separated string of all business key identifiers for the task. For example, SubmissionId#ApplicationId.
- 3. Write business indicator extractor class for new human task by implementing com.ofss.fc.app.adapter.workflow.task.ITaskBIExtractor.
- 4. Override WorkflowTaskConfiguration preference in config/preferences.xml and add constant for task specific business indicator extractor.

BIEXTRACTOR\$NewHumanTaskName = "<Fully qualified path of task specific business indicator extractor class>"

- 5. Create new DTO extending com.ofss.fc.app.workflow.dto.bam.task.TaskMetricDTO and populate the same in business indicator extractor class. For example, NewHumanTaskMetricDTO.
- 6. Create a data object class corresponding to NewHumanTaskMetricDTO, for example, NewHumanTaskMetricDO by extending abstract class com.ofss.fc.integration.analytics.bam.dataobject.framework.BAMDataObject.
- 7. Override "BAMDataObjectAssemblers" preference in config/preferences.xml and add constant for new data object assembler.

NewHumanTaskMetricDTO = "<Fully qualified path of data object assembler class>"

8. Create data object on BAM server in the Administration tab of BAM Composer.

4.2.2 Addition of custom business indicator(s) to existing data object

The following steps can be performed to add custom business indicators to existing OriginationTaskMetricDO data object.

- 1. Create new data object class, for example, CustomOriginationTaskMetricDO by extending existing com.ofss.fc.integration.analytics.bam.dataobject.banking.origination.OriginationTaskMetricDO and add new custom attributes.
- 2. Extend the existing implementation of task business indicator extractor class to populate dictionary object of OriginationTaskMetricDTO for new custom attributes.
- 3. Extend the existing implementation of data object assembler to populate and return instance of CustomOriginationTaskMetricDO.
- 4. Modify the definition of OriginationTaskMetricDO data object on BAM server using BAM architect tool to add custom business indicator attributes.

4.2.3 Publishing analytics data to system other than Oracle BAM

The following steps can be performed to publish analytics data to system other than Oracle BAM.

- 1. Write custom implementation class CustomRealtimeAnalyticsDataPublisher by extending existing class com.ofss.fc.app.adapter.impl.integration.analytics.RealtimeAnalyticsDataPublisher.
- 2. Write custom implementation class CustomAnalyticsDataPublisherFactory by implementing the interface com.ofss.fc.app.adapter.integration.analytics.IAnalyticsDataPublisherFactory and return the instance of CustomRealtimeAnalyticsDataPublisher class.
- 3. Override "AdapterFactories" preference in config/preferences.xml and override the constant value for "ANALYTICS_DATA_PUBLISHER_FACTORY".

ANALYTICS_DATA_PUBLISHER_FACTORY = "<Fully qualified path of CustomAnalyticsDataPublisherFactory class>"

4.2.4 Publishing new DataObject from OBP Domain

Any object from OBP Domain which extends DataTransferObject can be published to BAM server by following simple steps explained below.

- 1. Identify/Write the new CustomDataTransferObject that extends existing DataTransferObject which needs to be published to BAM and has all the requisite attributes which need to be presented in BAM.
- 2. Write the new CustomDataObject class by extending existing class BAMDataObject and adding desired flat attributes which are not complex types of OBP Domain. (Use Calendar for dates and BigDecimal for amounts). Also implement methods keyFieldsAsString() and getDataObjectPath() methods so as to return identical corresponding values as maintained for this DataObject in BAM.
- 3. Create new CustomDataObject in BAM through the Administration tab of BAM Composer, so as the names of the attributes are identical to those kept in CustomDataObject in OBP Domain.
- Create new CustomDataObjectAssembler class by implementing existing IDataObjectAssembler and overriding to BAMDataObject() method to return the CustomDataObject assembled from source CustomDataTransferObject.
- 5. Identify the preference for BAMDataObjectAssemblers from config/preferences.xml and in the preference, add corresponding entry for the new CustomDataTransferObject as key=value pair. Here, key is fully qualified class name string with dots replaced by underscores and value corresponds to the fully qualified name string of the CustomDataObjectAssembler class.
- For example, public final String com_example_CustomDataTransferObject = "com.example.assembler. CustomDataObjectAssembler";
- 7. Identify the events which will publish the CustomDataObject in the system flow inside OBP Domain. On such events, assemble the DataObject and invoke publish of existing BAMDataPublisher.

5 SOA Customizations

OBP provides the functionality for customizing the SOA composite applications. The steps to customize a SOA composite application are similar to those of customizing an ADF View Controller application with a few differences. The similarities and differences would be apparent in the examples demonstrated in the following sections.

The following section provides details about the SOA Components Customization. The detailed documentation for customizing and extending the SOA Components is also available at the Oracle website:

http://docs.oracle.com/cd/E25178_01/fusionapps.1111/e16691/ext_soaedit.htm

5.1 Customization Layer

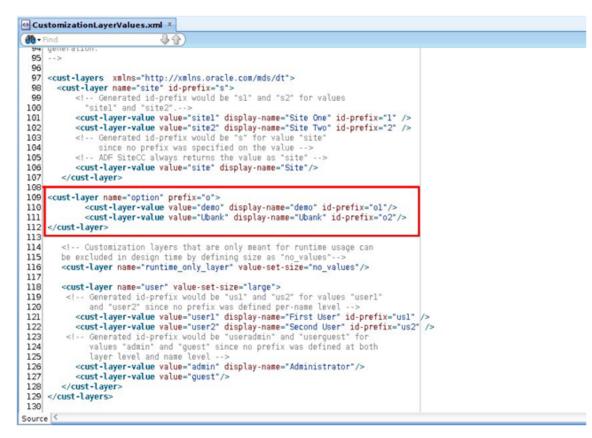
To customize an application, you must specify the customization layers and their values in the *CustomizationLayerValues.xml* file, so that they are recognized by JDeveloper.

You need to create a customization layer with name option and values demo and another bank name.

To create this customization layer, follow these steps:

- 1. From the main menu, choose the **File** -> Open option.
- 2. Locate and open the file *CustomizationLayerValues.xml* which is found in the *<JDEVELOPER_HOME>/jdeveloper/jdev* directory.
- 3. In the XML editor, add the entry for a new customization layer and values as shown in the following image.

Figure 5–1 Add an entry for new Customization Layer



4. Save and close the file.

5.2 Customization Class

Before customizing an application, a *customization* class needs to be created which is the interface that the *Oracle Meta-data Services* framework uses to define which customization layer should be applied to the application's base meta-data.

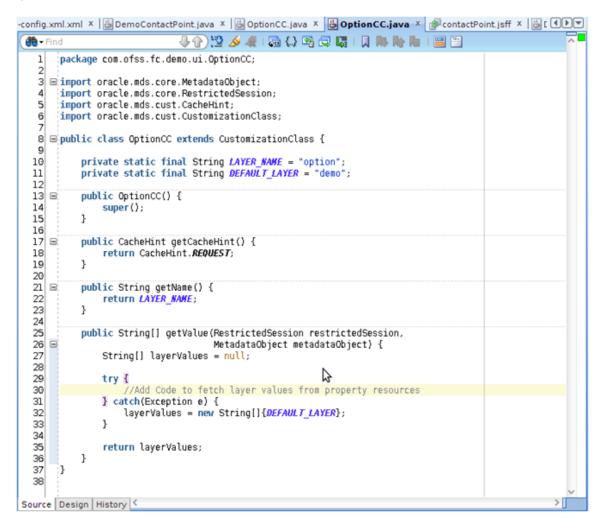
To create a customization class, follow these steps:

- 1. From the main menu, choose File -> New.
- 2. Create a generic project and give a name (com.ofss.fc.demo.ui.OptionCC) to the project.
- 3. Go to **Project Properties** for this project and add the required **MDS** libraries in the classpath of the project.
- 4. Create the customization class in this project. The customization class **must** extend the *oracle.mds.cust.CustomizationClass* abstract class.
- 5. Implement the following abstract methods of the CustomizationClass as follows:
 - getCacheHint() This method will return the information about whether the customization layer is applicable to all users, a set of users, a specific HTTP request or a single user.

- **getName()** This method will return the name of the customization layer.
- **getValue()** This method will return the customization layer value at runtime.

The screenshot below depicts a sample implementation of the above methods.

Figure 5–2 Create Customization Class



- 6. Build this class and deploy the project as a JAR file (com.ofss.fc.demo.ui.OptionCC.jar). This JAR file should only contain the customization class.
- 7. Place this JAR file in the location <*JDEVELOPER_HOME*>/*jdeveloper/jdev/lib/patches* so that the customization class is available in the classpath of Jdeveloper.

5.3 Enabling Application for Seeded Customization

Seeded customization of an application is the process of taking a generalized application and making modifications to suit the needs of a particular group. The generalized application first needs to be enabled for seeded customization before any customizations can be done on the application.

To enable seeded customization for the application, follow these steps:

- 1. Go to the Project Properties of the application's project.
- 2. In the ADF View section, check the Enable Seeded Customizations option.
- 3. In the *Libraries* and *Classpath* section, add the previously deployed com.ofss.fc.demo.ui.OptionCC.jar which contains the customization class.

Figure 5–3 Enabling Application for Seeded Customization

| Project Properties - , | /home/rshanbha/Work/jdev_workspace/Dec10/application/OriginationAndFulfillme | nt/process/com.ofss.fc.workflow.process.Notlfy |
|--|---|--|
| 🚯 Search | Libraries and Classpath | |
| E Project Source Paths E ADF Model | ○ Use <u>fu</u> stom Settings ⊙ <u>U</u> se Project Settings | Customige Settings |
| - ADF View | Java SE Version: | |
| Business Components | 1.6.0_29 (Default) | Change |
| Compiler | Classpath Entries: | |
| Dependencies Deployment | Export Description | Add Library |
| EJB Module | SOA Designtime SOA Runtime | Add JAR/Directory |
| Extension | Image: Solar Runtime Image: Solar Runtime | Bemove |
| Javadoc Java EE Application | Mediator Runtime | |
| JSP Tag Libraries | MDS Runtime DEMO OPTION CC | Edit |
| JSP Visual Editor | | Share As |
| Libraries and Classpath Resource Bundle | | Move Ug |
| - Run/Debug/Profile - Technology Scope | | Move Do <u>w</u> n |
| rechnology scope | | |
| Help | | OK Cancel |

- 4. In the **Application Resources** tab, open the *adf-config.xml* present in the *Descriptors/ADF META-INF* folder.
- 5. In the list of *Customization Classes*, remove all the entries and add the *com.ofss.fc.demo.ui.OptionCC.OptionCC* class to this list. The sections below will elaborate in detail the actual customization of a SOA process with examples.

5.4 SOA Customization Example Use Cases

This section describes the examples use cases of SOA customization.

5.4.1 Add a Partner Link to an Existing Process

In this example of SOA customization, we will be adding a Partner Link call to an Echo Service to an existing SOA process. The Echo Service will take a string input and respond with the same string as output.

The following section will explain how to create a SOA project and process with the example of Echo Service.

Step 1 Create SOA Project

You will need to create a SOA project to contain the Echo Service process. To create the SOA project, follow these steps:

- 1. In the Main Menu, go to File -> New.
- 2. In the Project Gallery that opens, select SOA Project and click OK.

Figure 5–4 Select SOA Project

| 🐞 Search | | | | |
|---|--|--|--|--|
| Categories: | Items: Show All Description | | | |
| | Extension Project Java Application Project | | | |
| B-Business Tier | 🗇 Java Project | | | |
| -ADF Business Components -Business Intelligence | D Project from Existing Source | | | |
| -Business Rules | Project from WAR File | | | |
| Data Controls EIB | Project Template | | | |
| Security — TopLink/JPA — Web Services — Client Tier — ADF Desktop Integration — ADF Swing — Extension Development | SOA Project Opens the Create SOA Project dialog, with which you create a service-oriented architecture project using new or existing service components like BPEL, Business Rules, Human Tasks, Mediator, Adapters, and Web Services. To enable this option, you must select an application, a project, or a file within a project in the Application Navigator. | | | |
| Swing/AWT | D TopLink Project | | | |
| Database Tier Database Files | Web Project | | | |

- 3. In the **Create SOA Project** wizard, enter appropriate project name (EchoService) and location for the project.
- 4. Click Next.

Figure 5–5 Enter SOA Project Name

| Project Name | Project Name: EchoService | |
|--------------|---|---|
| | Dirgctory: //home/rshanbha/Work/jdev_workspace/Dec10/application/Orig | ginationAndFulfillment/process/EchoService Browse |
| | Project Technologies Generated Components Associated Libraries | |
| | Available: | Selected: |
| | | SOA |
| | ADF Desktop Integration | |
| | ADF Faces | |
| | ADF Library Web Application Support | |
| | ADF Mobile Browser | |
| | | |
| | ADF Page Flow | |
| | ADF Swing | |
| | ADF Swing Ant | |
| | BPM | |
| | Database (Offline) | |
| | EJB | |
| | HTML | |
| | Java | |
| | JavaBeans | |
| | | |
| | Technology Description: | |

5. In the next dialogue of the wizard, enter appropriate name (EchoService) for the SOA composite.

- 6. Select *Empty Composite* from the drop-down menu.
- 7. Click Finish.

Figure 5–6 Configure SOA Settings

| Configure SOA settings | |
|--|-------------|
| Project Name EchoService | |
| EchoService | |
| | |
| Project SOA Settin Composite Template: | |
| Empty Composite | |
| Composite With BPEL Process | |
| Composite With Business Rule | |
| Composite With Mediator | |
| Composite With Human Task | |
| Composite With Spring Context | |
| Composite From Oracle BPA Blueprint | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| < Customizable | |
| | |
| Help < <u>Back</u> Next > Fir | nish Cancel |

Step 2 Add Mediator Component

You will need to add a *Mediator* component to the BPEL process to process the input to the SOA process and generate an output.

To add the *Mediator*, follow these steps:

- 1. From the **Project Navigator** tab, select and open *EchoService.bpel* in the *Design* mode.
- 2. From the Component Palette tab, in SOA Components section, select the Mediator component.
- 3. Drag and drop it onto the *bpel* process.
- 4. In the Create Mediator dialogue that opens, enter appropriate name (EchoService).
- 5. From the Templates drop-down, select Synchronous Interface.
- 6. Check the Create Composite Service with SOAP Bindings option.
- 7. Click OK.

Figure 5–7 Create Mediator

| ate Meo | liator | |
|---|--|--|
| | | |
| EchoSe | vice | |
| <table-cell-rows> Syn</table-cell-rows> | chronous Interface | • 3 |
| Crea | e Composite Service with SOAP Bindings | |
| Input | {http://xmlns.oracle.com/singleString}singleString | ٩ |
| Output: | {http://xmlns.oracle.com/singleString}singleString | Q |
|] | οκ | Cancel |
| | Compor a mediato EchoSer EchoSer Creat | Output: {http://xmlns.oracle.com/singleString}singleString |

- 8. An *EchoService.mplan* file will be created. Open this file in **Design** mode.
- 9. In the *Routing Rules* section, click the icon for *Add*.
- 10. Select Static Routing Rule from pop-up menu.
- 11. In the **Target Type** dialogue that opens, click **Echo**.

Figure 5–8 Select Target Type

| composite.xml × | EchoService.mplan × | 6 |
|---|--|---|
| ~8 | | |
| 4 Mediator | | |
| Name: WSDL URL: Port Type: Resequence Level: | EchoService EchoService.wsdl 🝓 execute_ptt operations 💌 | Should this routing rule invoke a service, trigger an event, or echo a response back to the initial caller? |
| Generations | les | Service Event Echo Cancel |
| 😂 execute | | Priority 4 😨 🗌 Validate Syntax (XSD) 🛆 🤝 💠 💥 |
| Callout To < | <java callout="" class="">></java> | 1 |
| | | |
| | | |
| | | |
| | | |
| Design Source Hist | orv | |
| teres frances frances | | |

A Static Routing section will be added to the screen.

- 12. Click the icon next to the *Transform Using* drop-down.
- 13. In the **Request Transformation Map** dialogue that opens, select the option **Create New Mapper File**.
- 14. Click **OK**.

| ame: | EchoService | | | | |
|---|-------------------------------|---------------|---|----|---------------------|
| SDL URL: | EchoService.wsdl | | | | |
| ort Type: | execute_ptt | | | | |
| esequence l | 😣 Request Transformat | ion Map | | | |
| | Transformation from request n | nessage req | uestMessage to message replyMessage. | _ | |
| Routine | Transformation to part r | eply | | | |
| Operation | Use Existing Mapper File: | | Q / | | ₽ ₽ ₽ |
| | Create New Mapper File: | singleString_ | To_singleString.xsl | | |
| 😫 execute | | | | | |
| Callout | Help | | OK Cancel | | |
| Static Rou | uting | | | | |
| < <fil< td=""><td>ter Expression>></td><td>880</td><td>Initial Caller*::execute:output</td><td>٩</td><td>Sequential •</td></fil<> | ter Expression>> | 880 | Initial Caller*::execute:output | ٩ | Sequential • |
| | Validate | Semantic | • | 8 | |
| | Transfo | rm Using r | eply : < <transformation map="">></transformation> | R | |
| | 1 m m | n Values | - | 00 | |

Figure 5–9 Request Transformation Map to create new mapper file

- 15. This will create a *singleString_to_singleString.xsl* file. Open this file in **Design** mode. You will see the input parameters in tree format on the left hand side and the output parameters on the right hand side of the screen.
- 16. In this case, the input and output contain a single *string*.
- 17. Select the input string from the left hand side and drag and drop it to the output string on the right hand side. This will create a mapping between input and output parameters.

| Source: EchoService.wsdl | | XSLT File: EchoService.wsd |
|--|--------------------------------------|----------------------------|
| - ⁽²⁾ <sources> └─ ⁽²⁾ inpl:singleString</sources> | | <target> 🔔 😑</target> |
| | 1 Date Fait Fait Fait Fait Fait Fait | |
| | | |
| | I LOID HONG HIND HIND HOLD IN | |
| | | |
| | | |
| | | |
| | | |

18. Save all files and build the project.

Step 3 Deploy Project to SOA Server

You will need to deploy this project to a SOA Server. From the *Admin* team, get details of the SOA Server and configure it in your JDeveloper.

After adding the SOA Server to your JDeveloper, follow these steps to deploy the *EchoService* composite to the server:

- 1. In the **Project Navigator** tab, right click the project and select *Deploy*.
- 2. In the Deploy EchoService dialogue that opens, select Deploy to Application Server from the list.
- 3. Click Next.

Figure 5–11 Select Deployment Action

| 😣 Deploy EchoSer | vice |
|----------------------------------|--|
| Deployment Action | |
| 🙊 Deployment Action | Select a deployment action from the list below. |
| Deploy Configuration Summary | Deploy to Application Server Deploy to SAR |
| < | Deploy this archive to SOA configured Application server(s) < Back |

4. In the Deploy Configuration dialogue, check the option *Overwrite any existing composites with the same revision ID*.

Figure 5–12 Deploy Configuration Settings

| 😣 Deploy EchoServ | ice | |
|---------------------|----------------------|---|
| Deploy Configuratio | n | |
| Deployment Action | Composite Revis | sion ID |
| Select Server | Project: | EchoService |
| O Summary | Current Revision ID: | 1.0 |
| | New Revision ID: | 1.0 |
| | 🗄 SOA Configurati | on Plan |
| | Keep running inst | sion as default. Ig composites with the same revision ID. ances on after redeployment. DA configuration plan for all composites: |
| | | Browse |
| Help | | < Back Next > Einish Cancel |

- 5. Click Next.
- 6. From the list of Application Servers, select the appropriate SOA Server.

Figure 5–13 Select Deployment Server

| 8 Deploy EchoSer | vice | |
|--|--|-----------------------------|
| Select Server | | |
| Deployment Action Deploy Configuration Select Server SOA Servers Summary | Application Servers: IntegratedWebLogicServer soa_server_mumbai_155 SOA_server_mumbai | දී 🛊 |
| < | ✓ Overwrite modules of the same name | < Back Next > Finish Cancel |

- 7. Click Next.
- 8. Select the appropriate *Partition* of the SOA Server where the composite should be deployed.

Figure 5–14 Select Target SOA Server

| OA Servers | | | | 010101010 | |
|----------------------|--------|---------------|------------|-----------|-----------------------------|
| C Deployment Action | Choose | | | | ant to deploy this archive. |
| Deploy Configuration | | SOA Server: | Partition: | Status: | Server URL: |
| Select Server | | 器 soa_serverl | default | RUNNING | http://ofsmug-vm-145.i |
| | | | | | |

9. Click Finish.

Step 4 Test Echo Service

After deploying the *EchoService* composite to a SOA Server, you can test it through the EM console:

- 1. Log in to *em* console of the SOA Server to which the composite is deployed.
- 2. From the SOA Domain select the EchoService composite.

Figure 5–15 Select SOA Domain

| Farm - | SOA Infrastructure - & Topology | | | | | | | | | |
|--------|---|----------------|---------------|------------------------|----------------|----------------|------------|-------------------|--------------------------|-------------------------|
| - | and hundred at street | ChoSer | vice [1.0 | 1.0 | | | | Looped in as | weblogic Ho | st OFSMUG-VM- |
| | | SOA Comp | | 10 | | | | ge Refreshed Dec | | |
| • | com.ofss.fc.workflow.process.DeclineOffer [1.0] | Sow comp | Jance + | | | | | ige werreshed Dec | 13, 2012 0.00. | 15 PH 041405.3 |
| | com.ofss.fc.workflow.process.EditSubmission [1.0] | | | | La La La La | 11-1 | | lie o | | |
| | com.ofss.fc.workflow.process.HumanTaskEventHandler | Running Insta | nces 0 To | tal 0 Active Retire | Shut Dow | en Test | Setting | s • 💁 🖻 | 1 | P Related Lini |
| | com.ofss.fc.workflow.process.InquirePartyDetails [1.0] | Dashboard | Instances | Faults and Rejected Me | rssages Unit T | ests Policies | | | | |
| | com.ofss.fc.workflow.process.InquiryPartyDetails [1.0] | 3 | | | | | | | | |
| | com.ofss.fc.workflow.process.ManageDepositOrigination | - | | | | | | | | |
| | com.ofss.fc.workflow.process.ManageLendingOrigination | Recent In | stances | | | | | | | |
| | com.ofss.fc.workflow.process.ManageOriginationFault [1 | Show Only I | tunning Insta | ances | Running 0 | | | Total O | | |
| | com.ofss.fc.workflow.process.ManageUserPreferences [| Instance ID | Name | Conversation | ID State | | | | | Start Time |
| | com.ofss.fc.workflow.process.NotifyCustomerHub [1.0] | No composite | | | | | | | | |
| | com.ofss.fc.workflow.process.OpenDemandDeposibAcco | | | | | | | | | |
| | com.ofss.fc.workflow.process.OpenTermDepositAccount | | | | | | | | | |
| | com.ofss.fc.workflow.process.OriginateDebitCard [1.0] | | | | | | | | | |
| | com.ofss.fc.workflow.process.OriginateJointApplicant [1.0] | | | | | | | | | |
| | com.ofss.fc.workflow.process.OriginateLending [1.0] | B Show All | | | | | | | | |
| | com.ofss.fc.workflow.process.OriginateLiability [1.0] | B Recent Fa | ults and R | ejected Messages | | | | | | |
| | com.ofss.fc.workflow.process.PerformCollateralPerfection | #1 | | | | | | | | |
| | com.ofss.fc.workflow.process.PerformCollateralValuatio | Show only sy | Jem rauros | æ | | | | | A | |
| | com.ofss.fc.workflow.process.PerformOriginationAndFul | Error Messag | je . | | Recovery | | Fault Time | Fault Location | Composite Instance II | Logs |
| | com.ofss.fc.workflow.process.PerformOriginationAndFul | No faults four | d | | | | | | | |
| | com.ofss.fc.workflow.process.PerformOriginationAndFul | | | | | | | | | |
| - | com.ofss.fc.workflow.process.PerformSettlement [1.0] | | | | | | | | | |
| - | com.ofss.fc.workflow.process.PerformSubmissionFraud | | | | | | | | | |
| • | com.ofss.fc.workflow.process.PerformTitleSearch [1.0] | | | | | | | | | |
| | com.ofss.fc.workflow.process.ProcessLendingApplication | | | | | | | | | |
| | com.ofss.fc.workflow.process.ProvisionIdentity [1.0] | | | | | | | | | |
| | com.ofss.fc.workflow.process.ReferDownsell [1.0] | 3 Show All | | | | | | | | |
| | com.ofss.fc.workflow.process.StructureDepositSolution | | | | | | | | | |
| | com.ofss.fc.workflow.process.StructureLendingSolution | Compone | nt Metrics | | | | | | | |
| | com.ofss.fc.workflow.process.VerifyApplication [1.0] | Name | | Compone | nt Type T | otal Instances | Running In | stances | Faulted Inst | |
| | com.ofss.fc.workflow.process.VerifyIdentity [1.0] | | | | | | | Re | | in Recoverable |
| | com.ofss.fc.workflow.process.WithdrawSubmission [1.0 EchoService [1.0] | EchoSe | Avice . | Mediator | | 0 | | 0 | 0 | 0 |
| | EchoService[1:0] | | | | | | | | | |
| | HelloWorld [1.0] | | | | | | | | | |
| | IPMBulkUploadProcess [1.0] | | | | | | | | | |
| | Project1 [1.0] | Services | and Refere | nces | | | | | | |
| | SampleSOA [1.0] | | | | | | | | | |
| | SampleSOA [2.0] | Name | | | Type | Usage | Faults | Total Message | Average P | rocessing Time (sec) |
| | Test [1.0] | S EchoSe | rvice_ep | | Web Service | Service | 0 | | 0 | 0.000 |
| | opic Domain | | | | | | | | | |
| | | | | | | | | | | |
| | ata Repositories | | | | | | | | | |

- 3. On the right hand side panel, you can see the *Dashboard* which lists the instances of SOA requests to that composite and many other options.
- 4. Click the **Test** button to test the composite.

Figure 5–16 Test Web Service

| ChoSer OA Compo | vice [1.0] () | | Logged in as weblogic Host OFS Page Refreshed Dec 13, 2012 6:08:55 PM |
|--------------------|--------------------|------------------------|---|
| ron compe | | | |
| est Web | Service 2 | | Test Web Serv |
| SDL. When | the page refreshes | with the WSDL details, | are not in the farm. To test a Web service, enter the WSDL and click Pars first select the Service, then select the Port, and then select the Operation and click Test Web Service. |
| WSDL | http://OFSMUG-VM | -145:8001/soa-infra/se | ervices/default/EchoService/EchoService_ep?WSDL |
| | Parse WSDL | | |
| | HTTP Basic Auth Op | tion for WSDL Access | |
| Service | EchoService_ep | | |
| Port | execute_pt | | |
| Operation | execute * | | |
| idpoint URI | L http://ofsmug-vm | 145:8001/soa-infra/se | ervices/default/EchoService/EchoService_ep Edit Endpoint URL |
| Request | Response | | |
| : Securit | Y | | |
| • Quality | of Service | | |
| | ransport Options | | |
| Addition | nal Test Options | | |
| Input A | rguments | | |
| Tree Viev | v V | | |
| | | Туре | Value |
| Name | | 1100 | 10.00 |

5. In the *Input Arguments* section, enter input and click *Test Web Service*. You will be able to see the response in the *Response* section.

Step 5 Add Customizable Scope to SOA Application

By default, a BPEL process in itself is not customizable. In addition to the steps followed to enable customizations in a SOA application, you will need to add a *Scope* component to the BPEL process and enable it for customizations.

To demonstrate customizations of a SOA process, we will be using the BPEL process *NotifyCustomerHubProcess* present in the composite *com.ofss.fc.workflow.process.NotifyCustomerHub*.

To see the flow of the *NotifyCustomerHubProcess* before customizations:

- 1. Deploy the composite to a SOA Server.
- 2. Log in to the *em* console and select the process from SOA Domain.
- 3. From the Dashboard, click Test.
- 4. Enter appropriate input and click Test Web Service.
- 5. From the Dashboard, click an Instance of the composite request.
- 6. Select the **Flow** tab to see the flow of the process.

| Flow Trace > Instance of NotifyCustomerHubProcess | Data Refreshed Dec 13, 2012 6:17:34 | PH GHT+05:30 (2) |
|---|-------------------------------------|------------------|
| | | |
| SInstance of NotifyCustomerHubProcess (a) | | |
| This page shows BPEL process instance details. 🕐 | Instance ID bpel:450 | |
| | Started Dec 13, 2 | 012 6:16:55 PM |
| Audit Trail Flow Sensor Values Faults | | |
| Click an activity to view the details. | Current Audit Level: production@ | View Raw XML |
| publishPartyD | | |

Figure 5–17 Customization of SOA Application - Flow

7. Open the SOA application which contains the base composite which will be customizing. The aforementioned process is present in the *OriginationAndFulfillment* application inside the *com.ofss.fc.workflow.NotifyCustomerHub project*.

To add a customizable scope to the BPEL process, follow these steps:

- 1. Open the *NotifyCustomerHubProcess.bpel* file in **Design** mode.
- 2. From the *Component Palette* panel on the right side, in the *BPEL Constructs* section, drag the *Scope* component and drop it on to the BPEL process as shown in the figure.
- 3. Double-click the component and enter appropriate name (EchoServiceScope) for the component.
- 4. Drag and drop the existing *Assign* component labeled *setTitle* on to the newly added *EchoServiceScope* component.

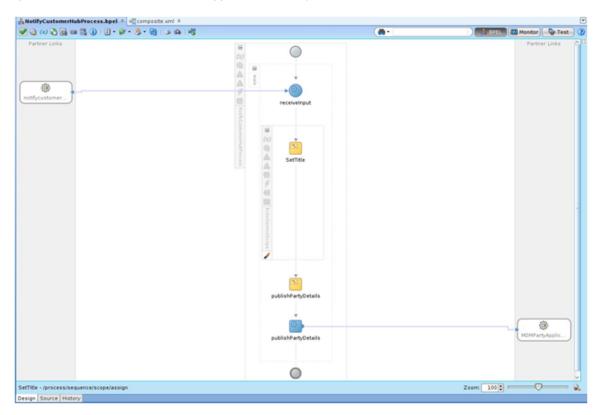


Figure 5–18 Customization of SOA Application - Notify Customer

- 5. Right click the *Scope* component and select *Customizable* from the context menu.
- 6. Save all the changes and restart JDeveloper in Customization Developer Role.

Step 6 Customize the SOA Composite

After adding a *Customizable Scope* to the base composite, you can start performing customizations in JDeveloper's *Customization Developer Role*.

When you open the *NotifyCustomerHubProcess.bpel* file in *Design* mode, you will notice that all other components in the process, except the customizable *EchoServiceScope* component, are disabled. This means that your customizations are limited to that scope.

In the following sections, we will be adding a *Partner Link* call to the previously created *EchoService* BPEL process and other required components in the *customization* mode.

Step 7 Add Partner Link Component

To add a *Partner Link* to the BPEL process, follow these steps:

- 1. From the Project Navigator, open the NotifyCustomerHubProcess.bpel file in Design mode.
- 2. From the *Component Palette* panel on the right side, in the *BPEL Constructs* section, drag the *Partner Link* component and drop it on to the *Partner Links* section of the BPEL process.
- 3. In the *Create Partner Link* dialogue that opens, enter appropriate name (EchoService) for the partner link.

- 4. In the WSDL Settings section of the dialogue, enter the URL for the previously created EchoService composite.
- 5. You will get alert notifying that there are no Partner Links defined in the current WSDL with an option to create a wrapper WSDL file with partner links defined for specified WSDL.
- 6. Click Yes. A new EchoServiceWrapper.wsdl file will be created which contains the partner links.
- 7. Select the newly defined partner link type and partner role in the *Partner Link Type* and **Partner Role** drop-down.
- 8. Select *Not Specified* option in the **My Role** drop-down.

Figure 5–19 Add Partner Link Component

| Application X BPM Project Navig X 🕞 X 😱 | NoblyCustomerHubProcess.bpel X | Component Palette × | G |
|--|---|--|---|
| 왕 📴 OriginationAndFulfillment 🔹 🔚 • | ✔ 🎱 🗶 🔁 💷 🚆 🐌 🗐 + 🕼 + 😓 + 🍓 🌫 🏟 + 🔤 🚮 Honitor) 🖓 Test. | BPEL 2.0 | |
| Image: Sol Control of Sol Control | Image Property Documentation Name: EchoService Pertner Link Image Partner Role NSDL Settings NSDL URL: /defaultitchoService./epTwSDL Image | | |
| | Partner Link Type: 5" Pagtner Role: By Role: Belo Belo Belo Concel Even: 2ve/0 Cancel Concel | a a a a a b c c c c c c c c c c c c c | |

Step 8 Add Invoke Component

You will need to add an *Invoke* component to invoke the previously added partner link call to *EchoService*.

To add Invoke component, follow these steps:

- 1. From the *Component Palette* panel on the right side, in the *BPEL Constructs* section, drag the *Invoke* component and drop it on the BPEL process inside the *EchoServiceScope* component.
- 2. Click the *Invoke* component and drag it to the previously added *EchoService* partner link.
- 3. Double-click the Invoke component.
- 4. In the *Edit Invoke* dialogue that opens, enter an appropriate name (invokeEchoService) for the component.
- 5. Click the icon for adding a new variable in the *Input Variable* and *Output Variable* sections.
- 6. Click **OK** to save the changes.

Figure 5–20 Add Invoke Component

| Documentation | Targets | Sources | Headers | Assertions | Skip Condition | |
|------------------|---------------|---------------|---------------|------------|----------------|-------------|
| General | ſ | Correlati | ions | Prop | perties | Annotations |
| Name: | invokeEch | oService | | | | |
| Conversation ID: | | | | | | |
| Detail Label: | | | | | | |
| | Invoke | as Detail | | | | |
| - Interaction T | | See Secondary | Ì | | | |
| | | | , | | | |
| Partner Link: | | | | | | <i>Q</i> |
| Port Type: | Texecute | _ptt | | | | - |
| Operation: | execute | | | | | - |
| Input Out | put | | | | | |
| O To Parts | Inpu | t Variable | | | | |
| Input: invok | EchoSenviz | a avaruta l | nnut/ariable | | | 49 |
| Tubar Interes | ereno servic | e_execute_ | inputvariable | , | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

Step 9 Add Assign Components

An *Assign* component is used to assign values to a variable. These values can be directly assigned from one variable to another or modified using BPEL functions available.

The *EchoService* accepts a single string as an input and gives a single string as an output. The *Input Variable* and *Output Variable* defined in the previously created *invokeEchoService* component will be used to hold the input value for the *EchoService* and the output returned respectively.

In our case, we will need to add two Assign components for following purposes:

- To populate the *Input Variable* of the *invokeEchoService* component with the value returned by the existing *setTitle* component.
- To populate the setTitle component with the value returned in the Output Variable of the invokeEchoService component.
- To add the Assign components

To add the Assign components, follow these steps:

- 1. From the *Component Palette* panel on the right side, in the *BPEL Constructs* section, drag the *Assign* component and drop it on the BPEL process inside the *EchoServiceScope* component between the *setTitle* and *invokeEchoService* components.
- 2. Double-click the Assign component.
- 3. In the *Edit Assign* dialogue that opens, enter appropriate name (copyToEchoServiceInput) for the component.
- 4. In the **Copy Rules** tab, select the *compositeInstanceTitle* from the left hand side tree and drag it to the *invokeEchoService_inputVariable* on the right hand side screen as shown in the figure.
- 5. Click OK.

Figure 5–21 Edit Copy Rules Variable

| General CopyRules Annotations Documentation Targets | Sources Skip Condition | |
|---|------------------------|---|
| Insert New Rule After 💌 | | 🗒 🚖 🔕 🌆 |
| NotifyCustomerHubProcess.bpel Partner Links Variables Variables Variables Variables Variables Variables (Variables) (Variable | Drag objects here | NotifyCustomerHubProces Partner Lin Variable Process UtVariable client NotifyCustomerHubProcessRequestMessage (X) + hPartyOetails_inputVariable ns1:publishPartyOetails400MParty (X) + is_OutputVariable ns1:publishPartyOetails400MParty (X) + compositeInstanceTitle xod string (X) + noService_execute_InputVariable option_ns1:requestMessage (X) + ns5:singleString string (X) + choService_execute_OutputVariable option_ns1:replyMessage (X) + |
| From | То | + × ≙ ⇒ |
| 🖥 🖬, \$compositeInstanceTitle | 🖬 \$invokeEc | hoService_execute_InputVariable.request |

6. Repeat the above steps for another *Assign* component labeled *copyFromEchoServiceOutput*. This component should be present after the *invokeEchoService* component.

The Copy Rules for this component should be as shown in the figure below.

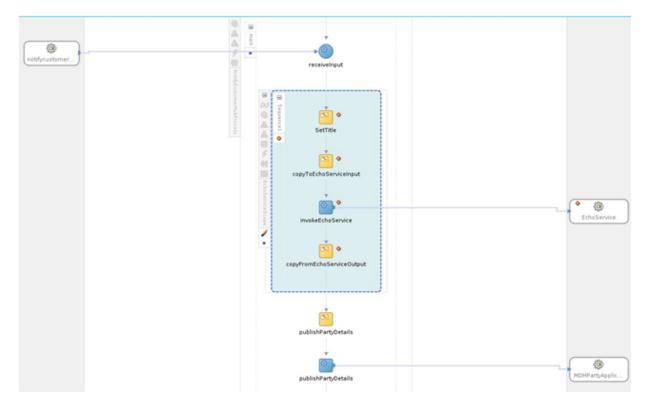
Figure 5–22 Add Assign Components - Reply

| | Sources Skip Condition |
|---|---|
| insert New Rule After 💌 | 📆 🚖 🔾 🌆 🤅 |
| NotifyCustomerHubProcess.bpel Partner Links Variables Variables () InputYariable client.NotifyCustomerHubProcessReport () () publishPartyOetals_ublishPartyOetals_inputYariable () publishPartyOetals_publishPartyOetals_outputYariable () () publishPartyOetals_publishPartyOetals_outputYariable () () publishPartyOetals_publishPartyOetals_outputYariable () () () () () () () () () () () | Details_inputVariable no1_publishPartyDetailsMDHParty (x) + putVariable no1_publishPartyDetailsMDHPartyResponse (x) + compositeInstanceTitle xsd(string (x) - |
| · · · · · · · · · · · · · · · · · · · | × * * |
| | + 🗙 🕆 |
| | То |
| From | |

7. Save all the changes.

The **Design** view of the BPEL process should look as shown in the figure below:

Figure 5–23 Design View of the BPEL Process



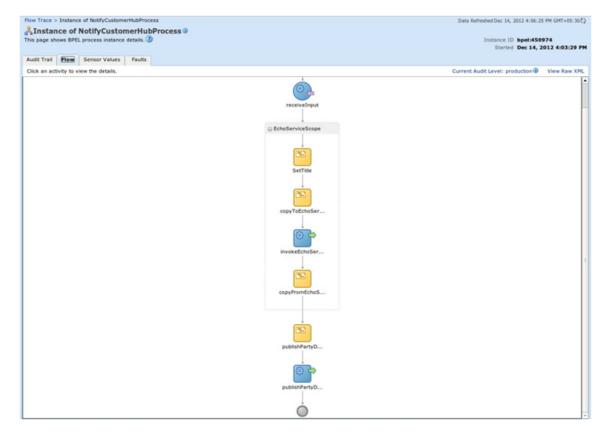
Step 10 Test the Customized Composite

After performing the customizations, build the project and deploy it to a SOA Server. You will need to include the *Customization Class JAR* in the runtime classpath of the deployed application.

To test the customized composite, follow these steps:

- 1. Log in to the em console and select the composite from the SOA Domain.
- 2. Click **Test** and enter appropriate input.
- 3. On the *Dashboard* panel, click the composite *Instance*. In the *Flow* panel of the screen, you will be able to see the flow of the customized composite.

Figure 5–24 Test Customized Composite - Flow



4. Click the *invokeEchoService* component from the flow to see the request and response XML for the invoke operation to the partner link.

Figure 5–25 Test Customized Composite - invokeEchoService

| | 4 16:03:45] rocation of operation "execute" on partner "EchoService". |
|--|--|
| [2012/12/ | 4 16:03:45] |
| Invoked 2 | way operation "execute" on partner "EchoService". |
| - < - <inv< td=""><td>ages> okeEchoService_execute_InputVariable> oart xmIns:xsi="http://www.w3.org/2001/XMLSchema-Instance" name="request"> <singlestring xmins="http://xmIns.oracle.com/singleString">000007918</singlestring> /part> rokeEchoService_execute_InputVariable> okeEchoService_execute_OutputVariable> okeEchoService_execute_OutputVariable> oart xmIns:xsi="http://www.w3.org/2001/XMLSchema-Instance" name="reply"> <inp1:singlestring xmins:inp1="http://xmIns.oracle.com/singleString">000007918 // and solve the solve</inp1:singlestring></td></inv<> | ages> okeEchoService_execute_InputVariable> oart xmIns:xsi="http://www.w3.org/2001/XMLSchema-Instance" name="request"> <singlestring xmins="http://xmIns.oracle.com/singleString">000007918</singlestring> /part> rokeEchoService_execute_InputVariable> okeEchoService_execute_OutputVariable> okeEchoService_execute_OutputVariable> oart xmIns:xsi="http://www.w3.org/2001/XMLSchema-Instance" name="reply"> <inp1:singlestring xmins:inp1="http://xmIns.oracle.com/singleString">000007918 // and solve the solve</inp1:singlestring> |
| <td><pre></pre></td> | <pre></pre> |

5.4.2 Add a Human Task to an Existing Process

This example demonstrates how to add a *Human Task* component mode.

In this example of SOA customization, we will be adding a *Human Task* to a BPEL process. Instead of adding the *Human Task* in customization mode, we will build a separate BPEL process with the human task and then customize the base composite to include a *Partner Link* call to that BPEL process.

The following section will demonstrate how to create a BPEL process with *Human Task*. The human task will take the title as a *string* input and will have the outcomes *REJECT* and *APPROVE*. The BPEL process will invoke the human task passing the title as input. Based on the outcome of the human task, the title will be suitably modified and returned by the BPEL process.

Step 1 Create SOA Project

You will need to create a SOA project to contain the Echo Service process. To create the SOA project, follow these steps:

- 1. In the Main Menu, go to **File -> New**.
- 2. In the Project Gallery that opens, select SOA Project.
- 3. Click Ok.

Figure 5–26 Select SOA Project

| 💏 Search | | |
|--|---|------------------------------------|
| ategories: | jtems: | Show All Description |
| | Extension Project Java Application Project | |
| Business Tier ADF Business Components | Java Project | |
| -Business Intelligence | Project from Existing Source | |
| -Business Rules -Data Controls | D Project from WAR File | |
| -EJB | - Roject Template | |
| -Security -TopLink/JPA -Web Services -Client Tier -ADF Desktop Integration -ADF Swing -Extension Development | SOA Project Opens the Create SOA Project dialog, with which you project using new or existing service components like Mediator, Adapters, and Web Services. To enable this option, you must select an application, the Application Navigator. | BPEL, Business Rules, Human Tasks, |
| -Swing/AWT | D TopLink Project | |
| Database Tier | Web Project | |

4. In the **Create SOA Project** wizard, enter appropriate project name (TitleApproval) and location for the project and click **Next**.

Figure 5–27 Create SOA Project Name

| 8 Create SOA Proj Name your project | ject - Step 1 of 2 |
|--|---|
| Project Name | Project Name: EchoService Dirgctory: /home/rshanbha/Work/jdev_workspace/Dec10/application/OriginationAndFulfillment/process/EchoService Browse Project Technologies Generated Components Associated Libraries |
| | Available: Selected: ADF Business Components ADF Desktop Integration ADF Faces ADF Veraces ADF Mobile Browser ADF Swing Ant BPPM Database (Offline) EjB HTML java JavaBeans |
| Kelp | Technology Description: SOA is the Service Oriented Architecture to build composite applications. < <u>Rack</u> <u>Next</u> > <u>Enish</u> Cancel |

- 5. In the next dialogue of the wizard, enter appropriate name (TitleApproval) for the SOA composite.
- 6. Select *Composite With BPEL Process* from the drop-down menu.

7. Click Finish.

Figure 5–28 Configure SOA Settings

| Configure SOA settings | | |
|------------------------|---|----------------------------|
| Project Name | Composite Name: TitleApproval | |
| Project SOA Settin | Composite Template: Empty Composite Composite With BPEL Process Composite With Business Rule Composite With Human Task Composite With Human Task Composite With Spring Context Composite From Oracle BPA Blueprint | |
| Help | <u>C</u> ustomizable | < Back Next > Finish Cance |

8. The dialog *Create BPEL Process* will open. Enter a name (TitleApprovalProcess) for the process and select *Asynchronous BPEL Process* from the templates drop-down.

Figure 5–29 Configure BPEL Process Settings

| 😣 Create | BPEL Pr | rocess | |
|---------------------|---------------------------------|--|----|
| BPEL Process | | | í, |
| | | service orchestration, based on the BPEL specification, used to describe/execute a business process (or large grained plemented as a stateful service. | |
| BPEL 1.1 Sp | ecificatio | on () BPEL 2.0 Specification | |
| <u>N</u> ame: | TitleApp | provalProcess | |
| Name <u>s</u> pace: | http://xn | mlns.oracle.com/OriginationAndFulfillment/TitleApproval/TitleApprovalProcess | |
| Template: | 중 Asynchronous BPEL Process · · | |) |
| Service Name: | titleappr | rovalprocess_client | |
| | Expos | se as a SOAP service | |
| | Delivery: | async.persist | - |
| | Input | {http://xmlns.oracle.com/OriginationAndFulfillment/TitleApproval/TitleApprovalProcess}process | 2 |
| | Output | {http://xmlns.oracle.com/OriginationAndFulfillment/TitleApproval/TitleApprovalProcess}processResponse | |
| | | | |
| Help | | OK Cancel | |

Step 2 Create Human Task

After defining the BPEL process, you will need to add the *Human Task* component to the process. To add the *Human Task*, follow these steps:

- 1. From the **Project Navigator** tab, select and open *composite.xml* in the *Design* mode.
- 2. From the **Component Palette** tab, in SOA Components section, select the Human Task component and drag and drop it onto the components section of the composite.xml.
- 3. In the **Create Human Task** dialog that opens, enter a name (TitleApprovalHumanTask) for the human task.
- 4. Click OK.

Figure 5–30 Enter Human Task Name

| 🛞 Creat | e Human Task | |
|---------------------|---|--------|
| | k task describes the tasks for users or groups to perform as part of an d business process flow. | 2 |
| <u>N</u> ame: | TitleApprovalHumanTask | |
| N <u>a</u> mespace: | http://xmlns.oracle.com/process/TitleApproval/TitleApprovalHumanTask | |
| | Create Composite Service with SOAP Bindings | |
| Help | ΟΚ | Cancel |

- 5. From *Project Navigator*, select and open *TitleApprovalHumanTask.task* file in **Design** mode. This file has the human task definition.
- 6. In the *General* section, specify a **Task Title** and **Description** for the human task.

Figure 5–31 Create Human Task - General Tab

| 🏠 General | | | |
|--|---|---|--|
| Data Assignment Presentation | Task Title: Description: | Text and XPath Title Approval Human Task Title Approval Human Task | |
| Deadlines Notification Access Events Documents | Outcomes: Priority: Category: Owner: Application Context: | APPROVE, REJECT 3 (Normal) By expression User Static Q | |

- 7. In the *Data* section, click the icon for add task parameter.
- 8. In the **Add Task Parameter** dialog, specify the parameter type and name for the input to the human task. In our case, the input task parameter would be a string title.

Figure 5–32 Add Human Task Parameter

| Define this parar | meter's type: | |
|-------------------|---|-----|
| 🚺 💿 <u>T</u> ype: | {http://www.w3.org/2001/XMLSchema}string | Q |
| | Includes standard simple XML types and types found in project schema | as. |
| ↔ ○ Element: | | 9 |
| | Define type by reference to elements found in project schemas. | |
| | Define type by reference to elements found in project schemas. | |
| Parameter Name | Define type by reference to elements found in project schemas. e: title Editable via worklist | |

Figure 5–33 Create Human Task - Data Tab

| 🚱 General | 😑 🔯 Data | | | | | | |
|----------------------------|----------|-------------------|-----------------|-------------|----------|--|--|
| Assignment Presentation | 1 | Name title | Element or Type | ema)string | Editable | | |
| Deadlines | 8 | Kapped Attributes | | | +/% | | |
| and a state of | | | 104655 | Description | | | |
| otification cess | | Label | Value | Description | | | |

The *Assignment* section is used to define the *Users* or *User Groups* to which the human task should be assigned.

- 9. Double-click Edit Participant.
- 10. In the Add Participant Type dialog, check the Value-based option for Specify Attributes Using.
- 11. Click the icon for adding a value.
- 12. Select the User By Name option and enter the name of your user (weblogic).

Figure 5–34 Add Participant Type Details

| pe: 🚺 Single | - | Label: | Stage1.Participant1 |
|--|-------------------|---------|------------------------|
| | | | e.g., Approval Manager |
| articipant List | | | |
| Build a list of particip | ants using: Names | and exp | pressions T |
| Build a list of particip | ants using: Names | and exp | pressions 🔻 |
| Build a list of particip Specify attributes u | | | ressions 💌 |
| Specify attributes u | | | |
| | | | tule-based |

Figure 5–35 Create Human Task - Assignment Tab

| 🚱 General 🍑 Data | 🗱 Make Parallel 🖥 Make Serial 🙈 💝 💠 🥒 Edit 💥 | Task will go from starting to final participant |
|---------------------|--|---|
| Assignment | | |
| Presentation | | |
| Deadlines | | |
| > Notification | | |
| Access | | |
| Events | | |
| Documents | â | |
| | T | |
| | Stagel | |
| | Stagel.Participant | L |

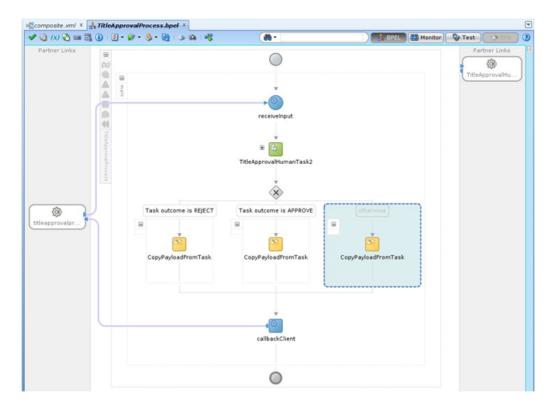
- 13. From the Project Navigator, open the *TitleApprovalProcess.bpel* file in **Design** mode.
- 14. From the **Component Palette**, select the component *Human Task* and drag-drop it on to the BPEL process.
- 15. Click the icon to add task parameter.
- 16. In the **Task Parameters** dialog, select the *string* input to the BPEL process.

Figure 5–36 Select Human Task Parameters

| Type: Variable | |
|---|--|
| | |
| > Variables | |
| Process | |
| 🖶 🗁 Variables | |
| inputVariable client: TitleApprovalProcessRequestMessage | |
| 🖻 📑 payload | |
| B- Client process <anonymous></anonymous> | |
| client input string | |
| ⊕-(x) outputVariable client:TitleApprovalProcessResponseMessage ⊕-[3] Scope - HumanTask1 | |
| Show Detailed Node Information | |
| | |
| Path: /client:process/client:input | |
| | |

17. In the task outcomes *Switch* in the BPEL process, delete the condition for *otherwise*.

Figure 5–37 Create Human Task - Delete Condition



- 18. From the **Component Palette**, select the *Assign* component and drag-drop it to the *REJECT* outcome of the switch.
- 19. Enter a *name* (rejectTitle) for the component.

20. In the *Copy Rules* section of the assign, use the *Expression Builder* to set the output *string variable* of the BPEL process to '<title> - Rejected'.

Figure 5–38 Create Human Task - Expression Builder

| 😣 Expression Builder | |
|--|---|
| Build an expression by typing directly into the Expression fragments from the fragment editors below the Expressio | field, using Ctrl+Space for XPath assistance, and/or insert n field. |
| Expression: | Ø 🕲 🗋 |
| concat(bpws:get/ariableData('input/ariable','payload','/cli | ent:process/client:input'), ' - Rejected') |
| 🔺 Insert In | to Expression |
| BPEL Variables | Functions |
| Variables | String Functions |
| Content <u>P</u> review: bpws:getVariableData('inputVariable', 'payload', '/client:pro | cess/client:input') |
| Description: Variable XPath expression | |
| Help | OK Cancel |

Figure 5–39 Create Human Task - Copy Rules

| Insert New Rule After 💌 | | 📆 🚖 🔕 連 🖷 |
|---|-------------------|---|
| valProcess.bpel r Links les cess Variables → ☐ payload → ۞ client.process <anonymous> ↓ ۞ client.process <anonymous> ↓ ۞ client.process <anonymous> ↓ ۞ client.ritleApprovalProcess (x) outputVariable client.TitleApprovalProcess (x) TitleApprovalHumanTask1_globalVariable</anonymous></anonymous></anonymous> | Drag objects here | TitleApprovalProcess.bpd Partner Links Process Process inputVariable client:TitleApprovalProcessRequestMessage (x) = outputVariable client:TitleApprovalProcessRequestMessage (x) = payload = client:processResponse <anonymous> client:processResponse <anonymous> c</anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous></anonymous> |
| < >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> | | To XPath: /client.processResponse/client.result |
| From From From | | То |

21. Save all changes to the human task. The BPEL process should look as shown in the figure below.

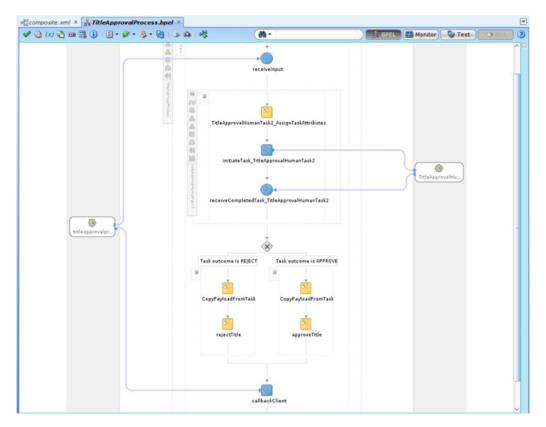


Figure 5–40 Create Human Task - BPEL Process

22. Deploy the SOA process to the server as mentioned in the previous example.

Step 3 Create Human Task Form

The *Human Task* is visible to assigned users in the *BPM Worklist* application. To display the task parameters and payload for the task, you will need to create a *task-flow* with the *Human Task* Form. This task form can be auto generated through the process. Follow these steps:

- 1. From the Project Navigator, open the *TitleApprovalHumanTask.task* file in **Design** mode.
- 2. Click the button for Create Task Form.
- 3. Select the Auto-generate Task Form option from the context menu.

Figure 5–41 Select Human Task Form

| Auto-Genera | te Task Form 📐 | | |
|----------------------------|----------------|---------------------------|---|
| Launch Task Data | Form Wizard | Text and XPath | |
| Assignment Presentation | Description: | Title Approval Human Task | |
| Deadlines | Outcomes: | APPROVE,REJECT | • |
| Notification | Priority: | 3 (Normal) 💌 | |
| Access | Category: | By expression 🔻 | |
| Events Documents | Owner: | User 🔻 Static 🔍 Q | |

- 4. Enter a name (TitleApprovalHumanTask) and *location* for the human task form project.
- 5. Click Finish.

The generated human task form project will have default file names. Re-Factor file names for form and page definition using appropriate naming conventions.

Step 4 Deploy Human Task Form Project

You will need to deploy the human task form project on the UI server for the previously deployed SOA process. To deploy, follow these steps:

- 1. Clean and build the project.
- 2. Right-click the project and select *Deploy* from the context menu.
- 3. In the **Deploy** dialog that opens, select the *Deploy to Application* Server option from the list and click *Next*.
- 4. Select the appropriate UI server for the SOA server.
- 5. Click Finish.

Figure 5–42 Select Human Task Form Deployment Action

| 8 | Deploy TitleApp | provalHumanTask |
|----|--|--|
| De | ployment Action | |
| R | Deployment Action | Select a deployment action from the list below. |
| | Select Server Weblogic Options Summary | Deploy to Application Server Deploy to WAR Create an archive and deploy it on a remote Application Server. |
| | Help | < Back Next > Finish Cancel |

Figure 5–43 Select Human Task Form - Weblogic Options

| 😣 Deploy TitleApp | provalHumanTask | | |
|---|--|--|---|
| Weblogic Options | | | |
| Deployment Action Select Server Weblogic Options Summary | O Deploy to <u>a</u> ll instances in | nces in the domain | Li <u>s</u> t All Servers |
| | Name AdminServer soa_server1 ui_server bam_server1 | Servers | Status RUNNING RUNNING RUNNING SHUTDOWN |
| | | be registered as a shared lit Application | |
| < | O Deploy as a shared Libr | | < Back Next > Finish Cancel |

Step 5 Add Customizable Scope to SOA Application

To demonstrate customizations of a SOA process, we will be using the BPEL process NotifyCustomerHubProcess present in the composite com.ofss.fc.workflow.process.NotifyCustomerHub.

Open the SOA application which contains the base composite which will be customizing. The aforementioned process is present in the *OriginationAndFulfillment* application inside the *com.ofss.fc.workflow.NotifyCustomerHub project*.

To add a customizable scope to the BPEL process, follow these steps:

- 1. Open the *NotifyCustomerHubProcess.bpel* file in **Design** mode.
- 2. From the *Component Palette* panel on the right side, in the *BPEL Constructs* section, drag the *Scope* component and drop it on to the BPEL process as shown in the figure.
- 3. Double click the component and enter appropriate name for the component.
- 4. Drag and drop the existing *Assign* component labeled *setTitle* on to the newly added *Scope* component.

Medify Cardenandinal Process Aged All Consects and All Co

Figure 5–44 Add Customization Scope to SOA Application

- 5. Right-click the Scope component and select Customizable from the context menu.
- 6. Save all the changes and restart JDeveloper in Customization Developer Role.

Step 6 Customize the SOA Composite

After adding a *Customizable Scope* to the base composite, you can start performing customizations in JDeveloper's *Customization Developer Role*.

When you open the *NotifyCustomerHubProcess.bpelf* ile in Design mode, you will notice that all other components in the process, except the customizable Scope component, are disabled. This means that your customizations are limited to that scope.

In the following sections, we will be adding a *Partner Link* call to the previously created *TitleApproval BPEL* process and other required components in the customization mode.

Step 7 Add Partner Link Component

To add a *Partner Link* to the BPEL process, follow these steps:

- 1. From the Project Navigator, open the NotifyCustomerHubProcess.bpel file in Design mode.
- 2. From the *Component Palette* panel on the right side, in the *BPEL Constructs* section, drag the Partner Link component and drop it on to the *Partner Links* section of the BPEL process.
- 3. In the *Create Partner Link* dialogue that opens, enter appropriate name (TitleApproval) for the partner link.
- 4. Select the following options:
 - a. TitleApprovalProcess as the Partner Link Type.
 - b. TitleApprovalProcessProvider as the Partner Role.
 - c. TitleApprovalProcessRequester as the My Role.

Figure 5–45 Add Partner Link Component

| Name: TitleAppro | oval |
|---|---|
| Process: | |
| 🔽 Initializ | e Partner Role |
| WSDL Settings | |
| | |
| | 🚯 I 🔊 🖉 |
| WSDL URL: | 🔍 🧟 ا 🔞 45:8001/soa-infra/services/default/TitleApproval/titleapprovalprocess_client_ep?WSDL |
| | 45:8001/soa-infra/services/default/TitleApproval/titleapprovalprocess_client_ep?WSDL |
| <u>W</u> SDL URL: Partner <u>L</u> ink Type: Pa <u>r</u> tner Role: | 45:8001/soa-infra/services/default/TitleApproval/titleapprovalprocess_client_ep?WSDL |

Step 8 Add Invoke Component

You will need to add an *Invoke* component to invoke the previously added partner link call to *TitleApproval*. To add *Invoke component*, follow these steps:

- 1. From the *Component Palette* panel on the right side, in the *BPEL Constructs* section, drag the *Invoke* component and drop it on the BPEL process inside the *Scope* component.
- 2. Click the Invoke component and drag it to the previously added TitleApproval partner link.
- 3. Double-click the Invoke component.
- 4. In the *Edit Invoke* dialogue that opens, enter an appropriate name (invokeTitleApproval) for the component.
- 5. Click the icon for adding a new variable in the *Input Variable* section.
- 6. Click **OK** to save the changes.

Figure 5–46 Add Invoke Component

| ocumentation | Targets | | leaders Asser | | tion | |
|-----------------|--------------------------|----------------|---------------|------------|------|-------------|
| General | | Correlat | ions | Properties | | Annotations |
| lame: | invokeTitle | Approval | | | | |
| onversation ID: | | | | | | |
| etail Label: | | | | | | |
| | Invoke a | as Detail | | | | |
| Interaction Ty | pe: 🔕 Pa | rtner Link - | | | | |
| Partner Link | TitleApprove | al | | | | Q |
| | | | | | | |
| | | rovalProcess | | | | - |
| Operation: | process | | | | | • |
| Input Outp | 2DC | | | | | |
| O To Parts | Inpu | t Variable | | | | |
| Input: invoke | eInput_proce | ess_InputVaria | ble | | | 💠 🔍 |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

Step 9 Add Receive Component

You will need to add a *Receive* component to receive output from the previously added partner link call to *TitleApproval*. To add *Receive* component, follow these steps:

- 1. From the *Component Palette* panel on the right side, in the *BPEL Constructs* section, drag the *Invoke* component and drop it on the BPEL process inside the *Scope* component.
- 2. Click the *Receive* component and drag it to the previously added *TitleApproval* partner link.
- 3. Double-click the *Receive* component.

- 4. In the *Edit Receive* dialogue that opens, enter an appropriate name (receiveTitleApproval) for the component.
- 5. Click the icon for adding a new variable in the Output Variable section.
- 6. Click **OK** to save the changes.

Figure 5–47 Add Receive Component using BPEL functions

| ocumentation Genera | | Sources Headers Correlations | Assertions | | Annotations | |
|------------------------|---------------|---------------------------------|------------|------------|-------------|---|
| Genera | | Correlations | | Properties | Annotations | |
| lame: | receiveTit | eApproval | | | | |
| onversation I | D: | | | | | |
| | Create | Instance | | | | |
| | | | | | | |
| - Interaction | Type: 🔯 Pa | rtner Link▼ | | | | |
| Partner Link | TitleApprova | 1 | | | | Q |
| Port Type: | TitleAppr | ovalProcessCallback | | | • | |
| Operation: | process! | | | | | |
| | | | | | | |
| O From Part | ts 💿 Variable | | | | | |
| Variable: re | ceiveOutput_p | rocessResponse_Input | Variable | | | Q |
| | | | | | | |
| | | | | | | |

Step 10 Add Assign Components

An *Assign* component is used to assign values to a variable. These values can be directly assigned from one variable to another or modified using BPEL functions available.

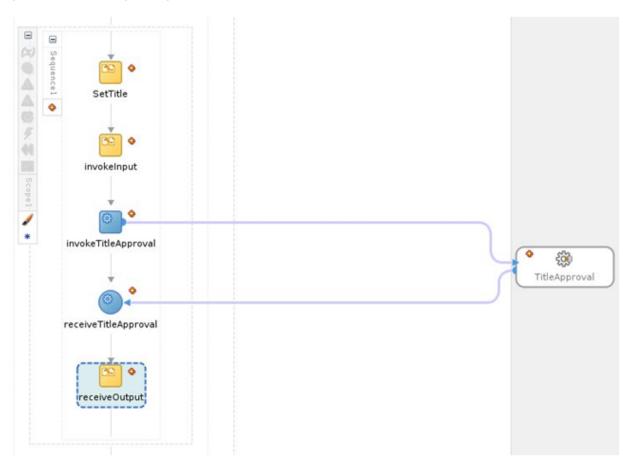
The *TitleApproval* accepts a single *string* as an input and gives a single *string* as an output. The *Input Variable* and *Output Variable* defined in the previously created *invokeTitleApproval* and *receiveTitleApproval* components will be used to hold the input value for the *TitleApproval* and the output returned respectively.

In our case, we will need to add two Assign components for following purposes:

- To populate the *Input Variable* of the *invokeTitleApproval* component with the value returned by the existing *setTitle* component.
- To populate the setTitle component with the value returned in the Output Variable of the receiveTitleApproval component.
- Add the two required *Assign* components and save all changes.

The customized process should look as shown in the figure below.

Figure 5–48 Add Assign Component



Step 11 Deploy and Test Customized SOA Composite

After performing the customizations, build the project and deploy it to a SOA Server. You will need to include the *Customization Class JAR* in the runtime classpath of the deployed application.

To test the customized composite, follow these steps:

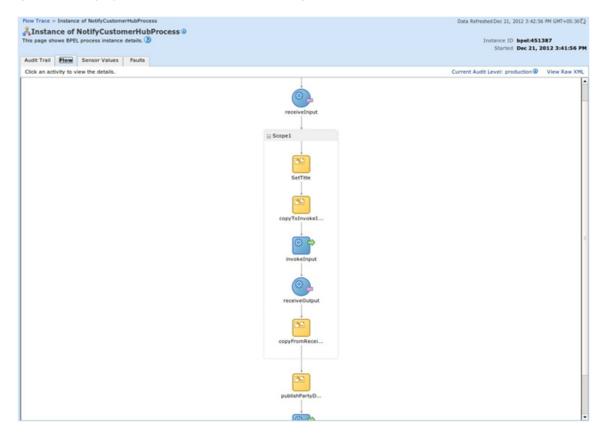
- 1. Log in to the *em* console and select the composite from the SOA Domain.
- 2. Click **Test** and enter appropriate input. A *Human Task* will be created and assigned to the user as specified in the human task definition.
- 3. Log in to the *BPM Worklist* application with the appropriate user. You will be able to see the previously created task on the dashboard.
- 4. Select the task from the list and click *Approve* or *Reject* button to perform approve or reject action on the task.

Figure 5–49 Deploy and Test Customized SOA Composite - My Tasks Tab

| Worklist Views 🔶 | 🗱 Hy Tasks Initiated Tasks Administrative Tasks |
|---|--|
| Inbox | Actions - 🚷 Assignee Me & Group * State Assigned * Search 🛞 Advanced |
| My Work Queues | Title Number Priority Assignees State Created Expires |
| Due Soon | Title Approval Human Task 201609 3 weblogic (U) Assigned Dec 21, 2012 10:5 |
| Past Day Past Week Past Week Past Quarter Past Quarter New Tasks My Views Shared Views | |
| and a state | |
| | Title Approval Human Task Actions - Approve Rev |
| Task Status Assigned | Title Approval Human Task Actions - Approve Rej |
| Task Status Assigned 1 Suspended | |
| Task Status Assigned 1 Suspended Completed 4 | Title Approval Human Task Actions - Approve Rej |
| Task Status Assigned 1 Suspended Completed 4 Expired | Title Approval Human Task Actions - Approve Rep Details Contents |
| Task Status Assigned 1 Suspended Completed Expired Withdrem | Title Approval Human Task Actions - Approve Rej |
| Task Status Assigned 1 Suspended Compileto 4 Expired Wil2drawn Akrted | Title Approval Human Task Actions - Approve Rej Details Contents Title 000007918 |
| Task Status Assigned 1 Suspended Completed Expired Wendrawn Alerted Errored | Title Approval Human Task Actions - Approve Rep Details Contents |
| Task Status Assigned 1 Suspended Completed 4 Expired Windrawn Alered | Title Approval Human Task Actions - Approve Rej Details Contents Title 000007918 |

- 5. On the *Dashboard* panel of the *em*, click the composite *Instance*.
- 6. In the *Flow* panel of the screen, you will be able to see the flow of the customized composite.

Figure 5–50 Deploy and Test Customized SOA Composite - Flow



7. Click the *invokeTitleApproval* component to see the request xml for the partner link call to *TitleApproval* process.

Figure 5–51 Deploy and Test Customized SOA Composite - Invoke Input



8. Click the *receiveTitleApproval* component to see the response xml for the partner link call to *TitleApproval* process.

Figure 5–52 Deploy and Test Customized SOA Composite - Receive Output

