



Informatica® ActiveVOS  
9.2.4.6

# 1. Getting Started

Informatica ActiveVOS 1. Getting Started  
9.2.4.6

March 2020

© Copyright Informatica LLC 1993, 2023

This software and documentation contain proprietary information of Informatica LLC and are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright law. Reverse engineering of the software is prohibited. No part of this document may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording or otherwise) without prior consent of Informatica LLC. This Software may be protected by U.S. and/or international Patents and other Patents Pending.

Use, duplication, or disclosure of the Software by the U.S. Government is subject to the restrictions set forth in the applicable software license agreement and as provided in DFARS 227.7202-1(a) and 227.7702-3(a) (1995), DFARS 252.227-7013(1)(ii) (OCT 1988), FAR 12.212(a) (1995), FAR 52.227-19, or FAR 52.227-14 (ALT III), as applicable.

The information in this product or documentation is subject to change without notice. If you find any problems in this product or documentation, please report them to us in writing.

Informatica, Informatica Platform, Informatica Data Services, PowerCenter, PowerCenterRT, PowerCenter Connect, PowerCenter Data Analyzer, PowerExchange, PowerMart, Metadata Manager, Informatica Data Quality, Informatica Data Explorer, Informatica B2B Data Transformation, Informatica B2B Data Exchange Informatica On Demand, Informatica Identity Resolution, Informatica Application Information Lifecycle Management, Informatica Complex Event Processing, Ultra Messaging, Informatica Master Data Management, and Live Data Map are trademarks or registered trademarks of Informatica LLC in the United States and in jurisdictions throughout the world. All other company and product names may be trade names or trademarks of their respective owners.

Portions of this software and/or documentation are subject to copyright held by third parties, including without limitation: Copyright DataDirect Technologies. All rights reserved. Copyright © Sun Microsystems. All rights reserved. Copyright © RSA Security Inc. All Rights Reserved. Copyright © Ordinal Technology Corp. All rights reserved. Copyright © Aandacht c.v. All rights reserved. Copyright Genivia, Inc. All rights reserved. Copyright Isomorphic Software. All rights reserved. Copyright © Meta Integration Technology, Inc. All rights reserved. Copyright © Intalio. All rights reserved. Copyright © Oracle. All rights reserved. Copyright © Adobe Systems Incorporated. All rights reserved. Copyright © DataArt, Inc. All rights reserved. Copyright © ComponentSource. All rights reserved. Copyright © Microsoft Corporation. All rights reserved. Copyright © Rogue Wave Software, Inc. All rights reserved. Copyright © Teradata Corporation. All rights reserved. Copyright © Yahoo! Inc. All rights reserved. Copyright © Glyph & Cog, LLC. All rights reserved. Copyright © Thinkmap, Inc. All rights reserved. Copyright © Clearpace Software Limited. All rights reserved. Copyright © Information Builders, Inc. All rights reserved. Copyright © OSS Nokalva, Inc. All rights reserved. Copyright Edifecs, Inc. All rights reserved. Copyright Cleo Communications, Inc. All rights reserved. Copyright © International Organization for Standardization 1986. All rights reserved. Copyright © ej-technologies GmbH. All rights reserved. Copyright © Jaspersoft Corporation. All rights reserved. Copyright © International Business Machines Corporation. All rights reserved. Copyright © yWorks GmbH. All rights reserved. Copyright © Lucent Technologies. All rights reserved. Copyright © University of Toronto. All rights reserved. Copyright © Daniel Veillard. All rights reserved. Copyright © Unicode, Inc. Copyright IBM Corp. All rights reserved. Copyright © MicroQuill Software Publishing, Inc. All rights reserved. Copyright © PassMark Software Pty Ltd. All rights reserved. Copyright © LogiXML, Inc. All rights reserved. Copyright © 2003-2010 Lorenzi Davide, All rights reserved. Copyright © Red Hat, Inc. All rights reserved. Copyright © The Board of Trustees of the Leland Stanford Junior University. All rights reserved. Copyright © EMC Corporation. All rights reserved. Copyright © Flexera Software. All rights reserved. Copyright © Jinfonet Software. All rights reserved. Copyright © Apple Inc. All rights reserved. Copyright © Telerik Inc. All rights reserved. Copyright © BEA Systems. All rights reserved. Copyright © PDFlib GmbH. All rights reserved. Copyright © Orientation in Objects GmbH. All rights reserved. Copyright © Tanuki Software, Ltd. All rights reserved. Copyright © Ricebridge. All rights reserved. Copyright © Sencha, Inc. All rights reserved. Copyright © Scalable Systems, Inc. All rights reserved. Copyright © jqWidgets. All rights reserved. Copyright © Tableau Software, Inc. All rights reserved. Copyright © MaxMind, Inc. All Rights Reserved. Copyright © TMate Software s.r.o. All rights reserved. Copyright © MapR Technologies Inc. All rights reserved. Copyright © Amazon Corporate LLC. All rights reserved. Copyright © Highsoft. All rights reserved. Copyright © Python Software Foundation. All rights reserved. Copyright © BeOpen.com. All rights reserved. Copyright © CNRI. All rights reserved.

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>), and/or other software which is licensed under various versions of the Apache License (the "License"). You may obtain a copy of these Licenses at <http://www.apache.org/licenses/>. Unless required by applicable law or agreed to in writing, software distributed under these Licenses is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the Licenses for the specific language governing permissions and limitations under the Licenses.

This product includes software which was developed by Mozilla (<http://www.mozilla.org/>), software copyright The JBoss Group, LLC, all rights reserved; software copyright © 1999-2006 by Bruno Lowagie and Paulo Soares and other software which is licensed under various versions of the GNU Lesser General Public License Agreement, which may be found at <http://www.gnu.org/licenses/lgpl.html>. The materials are provided free of charge by Informatica, "as-is", without warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose.

The product includes ACE(TM) and TAO(TM) software copyrighted by Douglas C. Schmidt and his research group at Washington University, University of California, Irvine, and Vanderbilt University, Copyright (©) 1993-2006, all rights reserved.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (copyright The OpenSSL Project. All Rights Reserved) and redistribution of this software is subject to terms available at <http://www.openssl.org> and <http://www.openssl.org/source/license.html>.

This product includes Curl software which is Copyright 1996-2013, Daniel Stenberg, <daniel@haxx.se>. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at <http://curl.haxx.se/docs/copyright.html>. Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

The product includes software copyright 2001-2005 (©) MetaStuff, Ltd. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at <http://www.dom4j.org/license.html>.

The product includes software copyright © 2004-2007, The Dojo Foundation. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at <http://dojotoolkit.org/license>.

This product includes ICU software which is copyright International Business Machines Corporation and others. All rights reserved. Permissions and limitations regarding this software are subject to terms available at <http://source.icu-project.org/repos/icu/icu/trunk/license.html>.

This product includes software copyright © 1996-2006 Per Bothner. All rights reserved. Your right to use such materials is set forth in the license which may be found at <http://www.gnu.org/software/kawa/Software-License.html>.

This product includes OSSP UUID software which is Copyright © 2002 Ralf S. Engelschall, Copyright © 2002 The OSSP Project Copyright © 2002 Cable & Wireless Deutschland. Permissions and limitations regarding this software are subject to terms available at <http://www.opensource.org/licenses/mit-license.php>.

This product includes software developed by Boost (<http://www.boost.org/>) or under the Boost software license. Permissions and limitations regarding this software are subject to terms available at [http://www.boost.org/LICENSE\\_1\\_0.txt](http://www.boost.org/LICENSE_1_0.txt).

This product includes software copyright © 1997-2007 University of Cambridge. Permissions and limitations regarding this software are subject to terms available at <http://www.pcre.org/license.txt>.

This product includes software copyright © 2007 The Eclipse Foundation. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at <http://www.eclipse.org/org/documents/epl-v10.php> and at <http://www.eclipse.org/org/documents/edl-v10.php>.

This product includes software licensed under the terms at <http://www.tcl.tk/software/tcltk/license.html>, <http://www.bosrup.com/web/overlib/?License>, <http://www.stlport.org/doc/license.html>, <http://asm.ow2.org/license.html>, <http://www.cryptix.org/LICENSE.TXT>, <http://hsqldb.org/web/hsqldbLicense.html>, <http://httpunit.sourceforge.net/doc/license.html>, <http://jung.sourceforge.net/license.txt>, [http://www.gzip.org/zlib/zlib\\_license.html](http://www.gzip.org/zlib/zlib_license.html), <http://www.openldap.org/software/release/license.html>, <http://www.libssh2.org>, <http://slf4j.org/license.html>, <http://www.sente.ch/software/OpenSourceLicense.html>, <http://fusesource.com/downloads/license-agreements/fuse-message-broker-v-5-3-license-agreement>, <http://antlr.org/license.html>, <http://aopalliance.sourceforge.net/>, <http://www.bouncycastle.org/licence.html>, <http://www.jgraph.com/jgraphdownload.html>, <http://www.jcraft.com/jsch/LICENSE.txt>, [http://jotm.objectweb.org/bsd\\_license.html](http://jotm.objectweb.org/bsd_license.html), <http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231>, <http://www.slf4j.org/license.html>, <http://nanoxml.sourceforge.net/orig/copyright.html>, <http://www.json.org/license.html>, <http://forge.ow2.org/projects/javaservice/>, <http://www.postgresql.org/about/licence.html>, <http://www.sqlite.org/copyright.html>, <http://www.tcl.tk/software/tcltk/license.html>, <http://www.jaxen.org/faq.html>, <http://www.jdom.org/docs/faq.html>, <http://www.slf4j.org/license.html>, <http://www.iodbc.org/dataspace/iodbc/wiki/IODBC/License>, <http://www.keplerproject.org/md5/license.html>, <http://www.toedter.com/en/jcalendar/license.html>, <http://www.edankert.com/bounce/index.html>, <http://www.net-snmp.org/about/license.html>, <http://www.openmdx.org/#FAQ>, [http://www.php.net/license/3\\_01.txt](http://www.php.net/license/3_01.txt), <http://srp.stanford.edu/license.txt>, <http://www.schneider.com/blowfish.html>, <http://www.jmock.org/license.html>, <http://xsom.java.net>, <http://benalman.com/about/license/>, <https://github.com/CreateJS/EaselJS/blob/master/src/easeljs/display/Bitmap.js>, <http://www.h2database.com/html/license.html#summary>, <http://jsoncpp.sourceforge.net/LICENSE>, <http://jdbc.postgresql.org/license.html>, <http://protobuf.googlecode.com/svn/trunk/src/google/protobuf/descriptor.proto>, <https://github.com/rantav/hector/blob/master/LICENSE>, <http://web.mit.edu/Kerberos/krb5-current/doc/mitK5license.html>, <http://jibx.sourceforge.net/jibx-license.html>, <https://github.com/lyokato/libgeohash/blob/master/LICENSE>, <https://github.com/hjiang/jsonxx/blob/master/LICENSE>, <https://code.google.com/p/lz4/>, <https://github.com/jedisct1/libsodium/blob/master/LICENSE>, <http://one-jar.sourceforge.net/index.php?page=documents&file=license>, <https://github.com/EsotericSoftware/kryo/blob/master/license.txt>, <http://www.scala-lang.org/license.html>, <https://github.com/tinkerpop/blueprints/blob/master/LICENSE.txt>, <http://gee.cs.oswego.edu/dl/classes/EDU/oswego/cs/dl/util/concurrent/intro.html>, <https://aws.amazon.com/ssl/>, <https://github.com/twbs/bootstrap/blob/master/LICENSE>, <https://sourceforge.net/p/xmlunit/code/HEAD/tree/trunk/LICENSE.txt>, <https://github.com/documentcloud/underscore-contrib/blob/master/LICENSE>, and <https://github.com/apache/hbase/blob/master/LICENSE.txt>.

This product includes software licensed under the Academic Free License (<http://www.opensource.org/licenses/afl-3.0.php>), the Common Development and Distribution License (<http://www.opensource.org/licenses/cddl1.php>), the Common Public License (<http://www.opensource.org/licenses/cpl1.0.php>), the Sun Binary Code License Agreement Supplemental License Terms, the BSD License (<http://www.opensource.org/licenses/bsd-license.php>), the new BSD License (<http://opensource.org/licenses/BSD-3-Clause>), the MIT License (<http://www.opensource.org/licenses/mit-license.php>), the Artistic License (<http://www.opensource.org/licenses/artistic-license-1.0>) and the Initial Developer's Public License Version 1.0 (<http://www.firebirdsql.org/en/initial-developer-s-public-license-version-1-0/>).

This product includes software copyright © 2003-2006 Joe Walnes, 2006-2007 XStream Committers. All rights reserved. Permissions and limitations regarding this software are subject to terms available at <http://xstream.codehaus.org/license.html>. This product includes software developed by the Indiana University Extreme! Lab. For further information please visit <http://www.extreme.indiana.edu/>.

This product includes software Copyright (c) 2013 Frank Balluffi and Markus Moeller. All rights reserved. Permissions and limitations regarding this software are subject to terms of the MIT license.

See patents at <https://www.informatica.com/legal/patents.html>.

DISCLAIMER: Informatica LLC provides this documentation "as is" without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of noninfringement, merchantability, or use for a particular purpose. Informatica LLC does not warrant that this software or documentation is error free. The information provided in this software or documentation may include technical inaccuracies or typographical errors. The information in this software and documentation is subject to change at any time without notice.

## NOTICES

This Informatica product (the "Software") includes certain drivers (the "DataDirect Drivers") from DataDirect Technologies, an operating company of Progress Software Corporation ("DataDirect") which are subject to the following terms and conditions:

1. THE DATADIRECT DRIVERS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT.
2. IN NO EVENT WILL DATADIRECT OR ITS THIRD PARTY SUPPLIERS BE LIABLE TO THE END-USER CUSTOMER FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL OR OTHER DAMAGES ARISING OUT OF THE USE OF THE ODBC DRIVERS, WHETHER OR NOT INFORMED OF THE POSSIBILITIES OF DAMAGES IN ADVANCE. THESE LIMITATIONS APPLY TO ALL CAUSES OF ACTION, INCLUDING, WITHOUT LIMITATION, BREACH OF CONTRACT, BREACH OF WARRANTY, NEGLIGENCE, STRICT LIABILITY, MISREPRESENTATION AND OTHER TORTS.

Publication Date: 2023-11-08

# Table of Contents

<b>Preface .....</b>	<b>5</b>
<b>Chapter 1: Resources.....</b>	<b>6</b>
Process Developer Components. ....	6
Process Developer Feature Tour. ....	7
XML Syntax. ....	8
Tips and Tricks. ....	9
BPMN Design Hints. ....	18
Glossary. ....	27
<b>Chapter 2: Introducing Business Process Execution Language.....</b>	<b>29</b>
BPEL for People Extension to WS-BPEL 2.0. ....	29
What is a BPEL Process. ....	30
BPEL Process Definition Elements. ....	31
Informatica Extensions to WS-BPEL 2.0. ....	32
<b>Chapter 3: ActiveVOS Tutorial.....</b>	<b>34</b>
Part 1: Starting a New Process. ....	35
Step 1: Create the Tutorial Orchestration Project. ....	36
Step 2: Create a New Process File in the Tutorial/bpel Folder. ....	38
Part 2: Planning and Designing a Process. ....	41
Step 1: Create a Receive activity. ....	42
Step 2: Working with Layout Features in the Process Editor. ....	42
Step 3: Save the File and Review BPEL Validation Messages. ....	43
Part 3: Creating a Process Service Consumer Participant. ....	43
Step 1: Viewing the WSDL and Schema. ....	44
Step 2: Using the Participants View to Create a Process Service Consumer. ....	44
Part 4: Creating Partner Service Provider Activities. ....	47
Part 5: Adding Process Activities and Properties. ....	51
Part 6: Adding Fault Handling. ....	64
Part 7: Adding Compensation and Correlation. ....	69
Part 8: Simulating the Process. ....	69
Part 9: Deploying the Process. ....	82
Part 10: Creating a Form to Run the Process. ....	89
Part 11: Debugging Your Process Remotely. ....	99
Part 12: Using the Web Services Explorer to Start a Process. ....	102

# Preface

This module contains information that about the features and components of Informatica ActiveVOS. This module contains information about the Business Process Execution Language (BPEL) that ActiveVOS and a tutorial.

# CHAPTER 1

## Resources

This chapter includes the following topics:

- [Process Developer Components, 6](#)
- [Process Developer Feature Tour, 7](#)
- [XML Syntax, 8](#)
- [Tips and Tricks, 9](#)
- [BPMN Design Hints, 18](#)
- [Glossary, 27](#)

## Process Developer Components

Process Developer includes the following components:

- **Process Developer**  
The Process Developer lets you create, simulate, deploy, and execute BPEL processes.
- **Orchestration Project Samples**  
The Process Developer tutorial and other orchestration samples contain files to get you started on basic and special processes.
- **Ant Runtime for Command Line Execution of Deployment and Testing Scripts**  
The installation wizard allows you to select the BUnit Runtime to install the Eclipse plugins necessary to run Process Developer scripts outside of Process Developer. These scripts include BPRD and BUnit Ant files. BPRD scripts include targets for deployments to the process Server. BUnit scripts include targets for unit testing your BPEL processes.

# Process Developer Feature Tour

Process Developer offers the following features:

<b>Simultaneously displayed diagrammatic and hierarchical view of process</b>	Create a process diagrammatically on the canvas. Use the synchronized Outline view to see a hierarchical element structure of the process. Source view is also available to view BPEL code that Process Developer generates.
<b>Process Editor canvas styles</b>	Select BPMN-centric, BPEL-centric BPMN, or Classic styles for modeling notation. Drag and drop icons onto a canvas to create a process. Process Developer creates valid BPEL code and generates a task list for missing and invalid activity properties.
<b>Participants and interfaces</b>	Catalog your business partners and WSDL and schema files in your project for automatic discovery and organization of all pertinent information stored in existing WSDL files. Comprehensive searching is available to locate namespaces, messages, and other elements. Use drag and drop operations to the Process Editor canvas for automated activity creation.
<b>Management of sample data</b>	Add sample data files for all WSDL messages for a convenient registry of test data across all processes using the messages. Add multiple files for various test scenarios. Automatically generate sample data for complex types. During simulation, test various execution paths using different data.
<b>Automatic static analysis</b>	Process Developer generates a problem list for all incomplete or invalid BPEL constructs so that you can fix problems without hunting for them. This feature works on imported BPEL files as well as native BPEL 1.1 and 2.0 files.
<b>Build a process by creating the participants</b>	Specify the process consumer and partner service roles for the process and then fill in the Web Service interaction activity details.
<b>Unit testing</b>	Comprehensive BUnit Editor and debugging environment for unit-testing process inputs and outputs.
<b>Automatic variable assignment</b>	Create Copy Operations automatically for new or existing Assign activities. Drag the Copy FROM variable to the Copy TO variable. Icons and colors indicate at a glance how a variable is used.
<b>Expression and query builders</b>	Process Developer gives you visual expression editing controls for building a wide range of scripts. In addition, you can readily extend the Process Developer expression editor to include your own expression language and custom functions. Built-in BPEL functions are automatically added to expressions.
<b>Activity properties</b>	Required and optional activity attributes are grouped for easy selection in the Properties view. Pertinent selections are in picklists. Add comments. Add correlation properties, compensation, and fault handling. Link to property definitions, such as WSDL operation.
<b>Create reusable BPEL components to re-use a selection of activities in other processes</b>	Select one or more activities on the Process Editor canvas and save them to the Custom Palette for later use. Significantly shorten design time by reusing modular elements.

<b>Simulation and debugging</b>	Simulate process execution using sample data. Set breakpoints, step through, or run the process. Remotely debug a process running on the server and suspend a process on an uncaught fault to perform exception management.
<b>Process deployment</b>	Deployment wizards help you provide endpoint references and policy assertions for services used in your process. A process deployment descriptor provides error-free techniques for binding your services.  Processes are automatically deployed to the appropriate server location within a package that contains all required files. The files become a contribution unit on the server and are managed as a unit.

You may also want to use the Process Developer tutorial to explore basic Process Developer concepts.

## XML Syntax

The following table lists XML symbols and their meaning for Business Process Execution Language (BPEL) XML syntax. The symbols indicate the allowable code according to the BPEL schema.

Symbol	Meaning	Example
+	One or more of this element is allowed	<code>&lt;correlation set="ncname" initiate="yes no"?&gt;+</code>
?	Zero or one element is allowed	<code>abstractProcess="yes no"?</code>
*	Zero or many of this element is allowed	<code>&lt;onAlarm (for=duration-expr   until="deadline-expr")&gt;*</code>

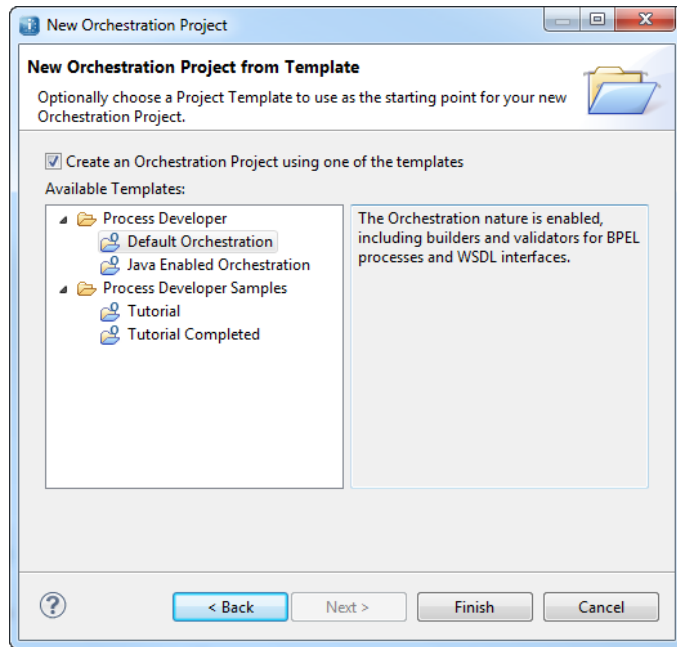


## Tips and Tricks

### Create an Orchestration Project to hold all resources

An orchestration project contains a collection of folders that Process Developer uses to build processes. Builders and validators make sure you're on the right track as you create processes.

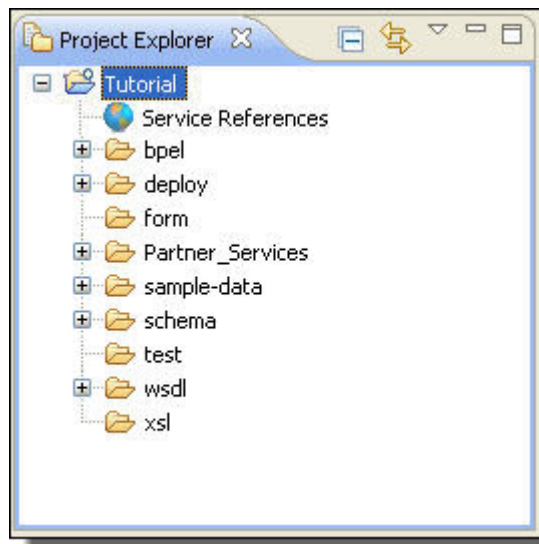
Start with a Process Developer Sample project.



### Jump Start a new BPEL process with service references and participants

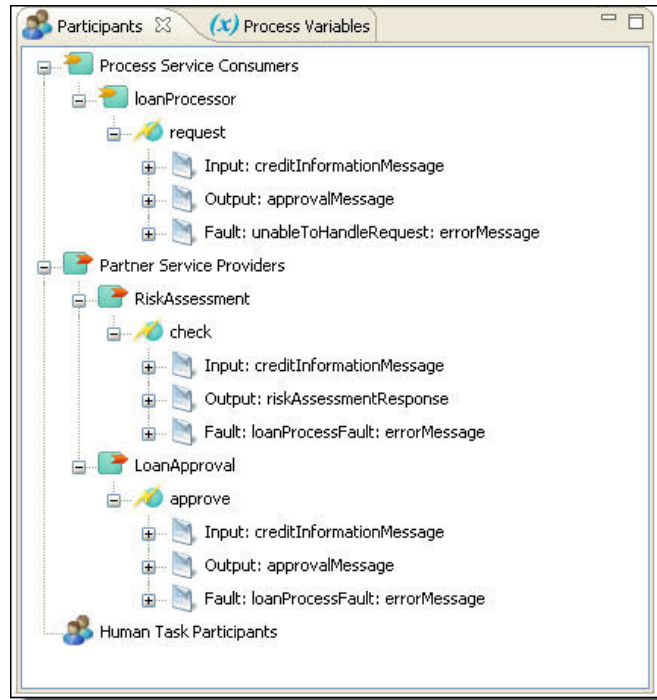
Add a WSDL file to service references or directly to an orchestration project before creating BPEL activities.

Process Developer catalogs WSDL files for automated activity creation.



### Participants

Define the Web services that use the process and the partner services that the process uses. This is a fast and intuitive way to start a process.



### Create activities for participants

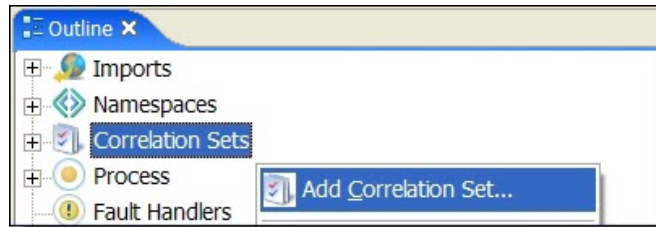
Drag a participant's operation to the Process Editor canvas to create activities.

Activities are contained within a sequence, so that adding new activities in sequence is easy.

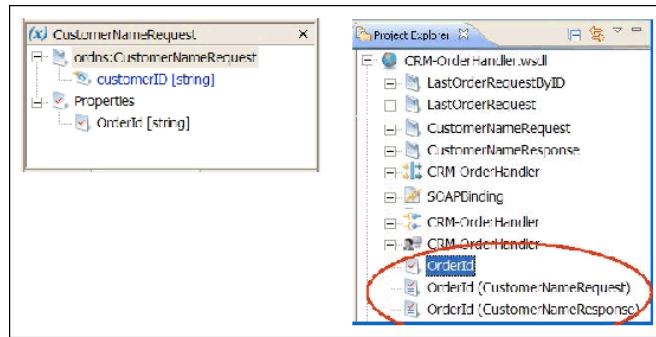


### Create message properties and property aliases

Message properties are WSDL extensions for BPEL that allow you to correlate messages for a long-running, asynchronous process. Select Add Correlation Set to create message properties and property aliases.

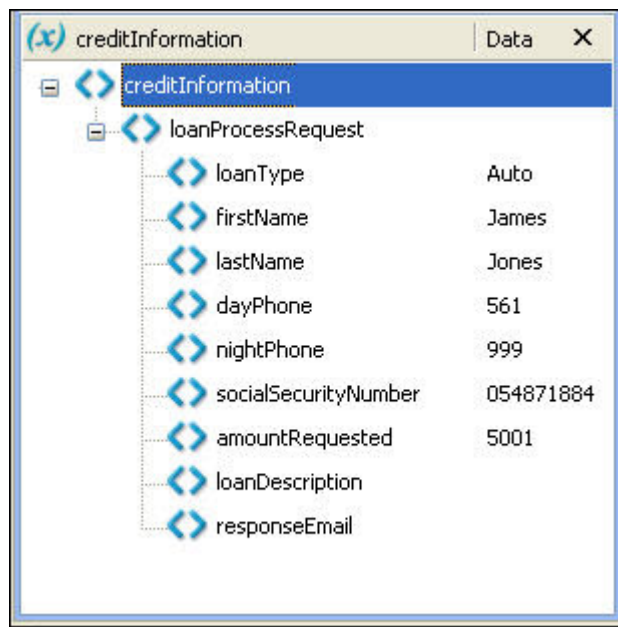


After you create message properties and property aliases, they are added to process variables and WSDL files. Next, you will create correlation sets, adding them to each activity that should correlate messages.



#### Use the convenient repository of sample data with automatic generation for complex types

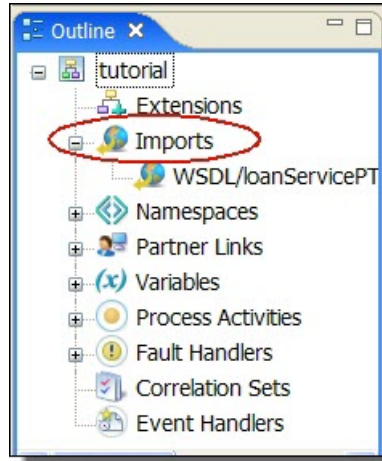
After entering Variables View, right-mouse click on a message part to generate or add sample data, instance documents for a complex type, or values for a simple type. For a complex part, Generate Sample automatically. Data is automatically loaded into process variables when you are simulating process execution. Data can be reused with other processes.



### Manually add a target namespace or other imports to a BPEL process

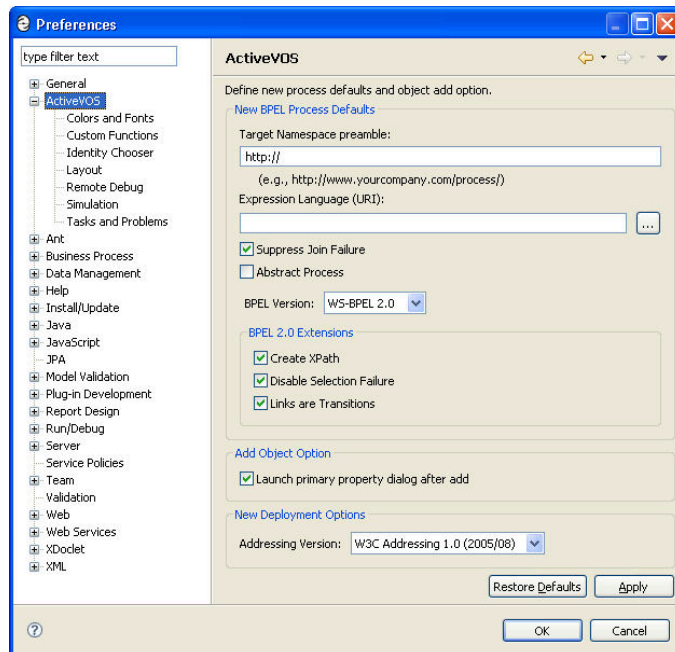
If you start a process with an operation, the target namespace is automatically added.

Manually add a target namespace in the Outline view by right-mouse clicking **Imports** and selecting **Add Import**.



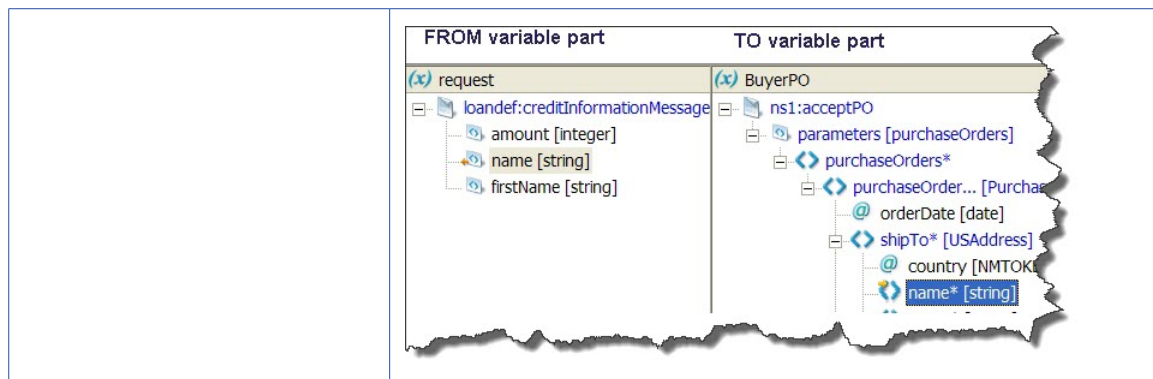
### Customize process settings

Process Developer has extensive settings for globally customizing BPEL processes. Set preferences for all processes, such as suppressing join failures and providing a default target namespace. Override the defaults for individual processes as needed.



### Create variable assignments automatically

Drag a like-typed process variable or variable part onto another process variable to automatically create a Copy operation. Blue color and from and to arrows differentiate variable parts.

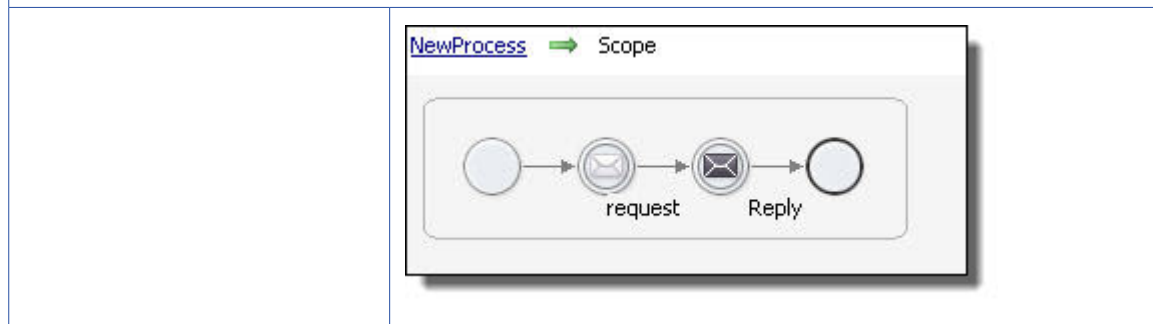


### Process viewing tips

#### *Collapsing Containers*

Manage screen real estate by collapsing containers like scopes, whiles, and sequences. Double-click to view the expanded activity in a drill-down window.

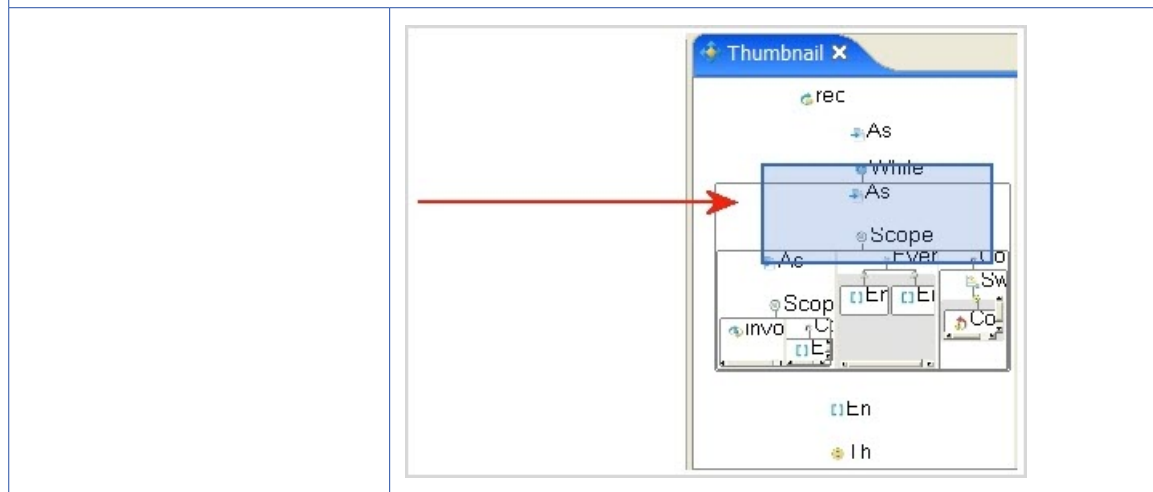
Make any activity horizontal or vertical.



#### *Thumbnail View*

Use Thumbnail view to select and pan a section of the process.

To display the Thumbnail view, select Window > Show View > Thumbnail



#### *Viewing full screen*

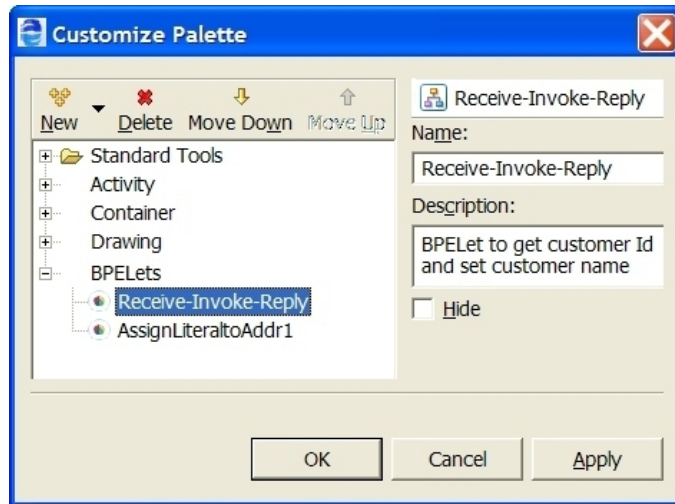
View a process full screen by double-clicking the process's title bar.

To view process properties and to activate the main toolbar, click anywhere on the Process Editor canvas to put it in focus.



### Significantly shorten design time with BPELets

Save any BPEL activity or set of activities as a BPELet-a custom activity. Drag a BPELet from the Custom palette to the canvas for any process.



### Discover and fix problems with automatic validation

View a list of errors, warnings, and information for invalid activities. Correct the problem, and it automatically disappears.

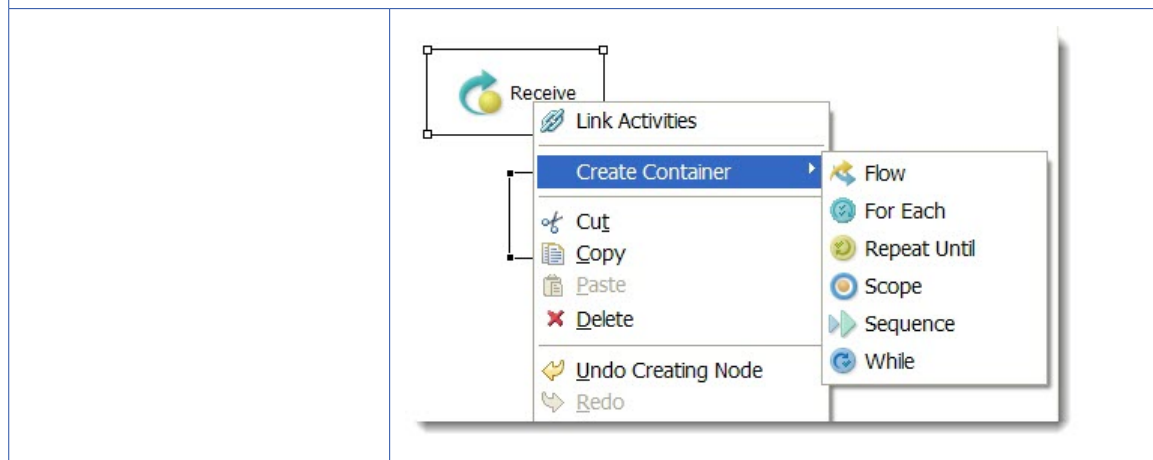


### Activity creation tips

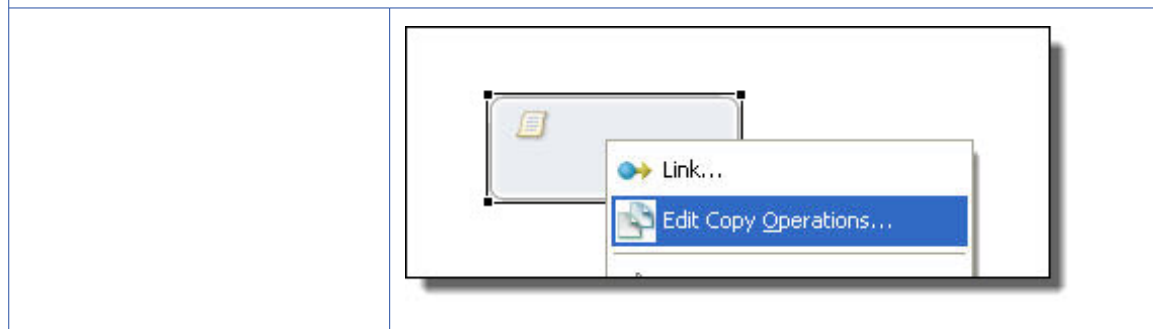
Add a link between activities by selecting the source activity and then selecting the target. Select Link Activities from the right mouse menu or use the toolbar Edge icon.



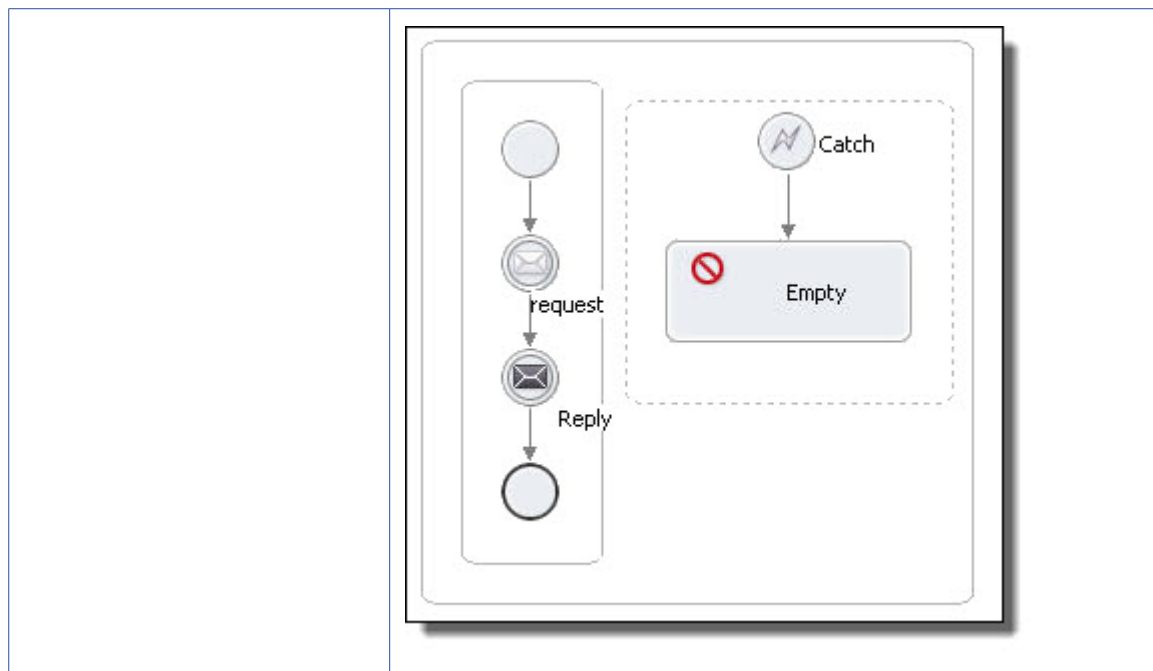
Contain a group of activities in a container by selecting them and then right mouse clicking on Create Container.



Add data mappings in activities like Receives and Replies. Alternately, add an Assign then add/edit copy operations in the Assign by double-clicking the activity or by selecting from the right mouse menu of the activity.



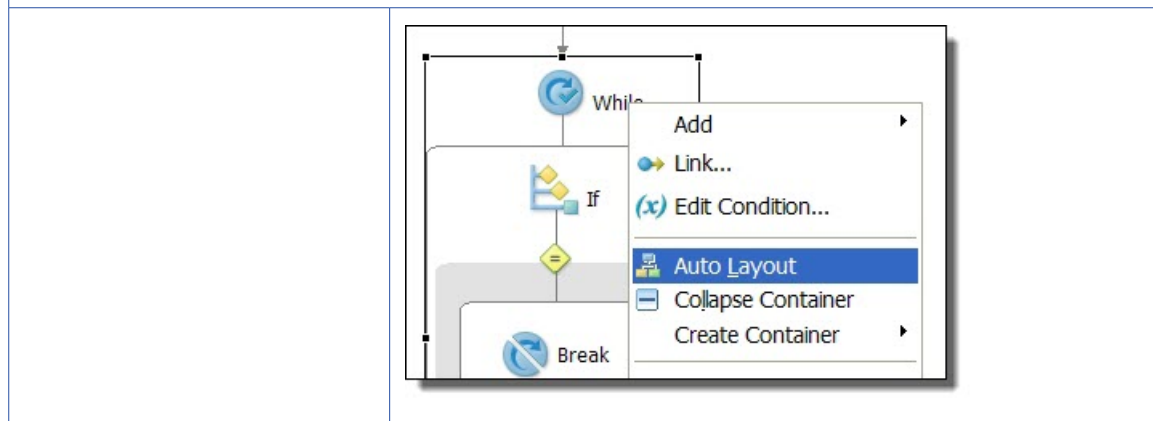
Add fault handling (or other handlers) to a scope by dragging a catch to a scope from the palette.  
Note that a fault activity is added to a catch container.



#### Process design tips

Use Auto Layout on any container, such as a While or If activity, to optimize your display area. You can also select a group of objects for auto layout.

Objects snap to a grid. Move an object one pixel at a time by using the Alt key + mouse combination.



Activity names can be long and descriptive.

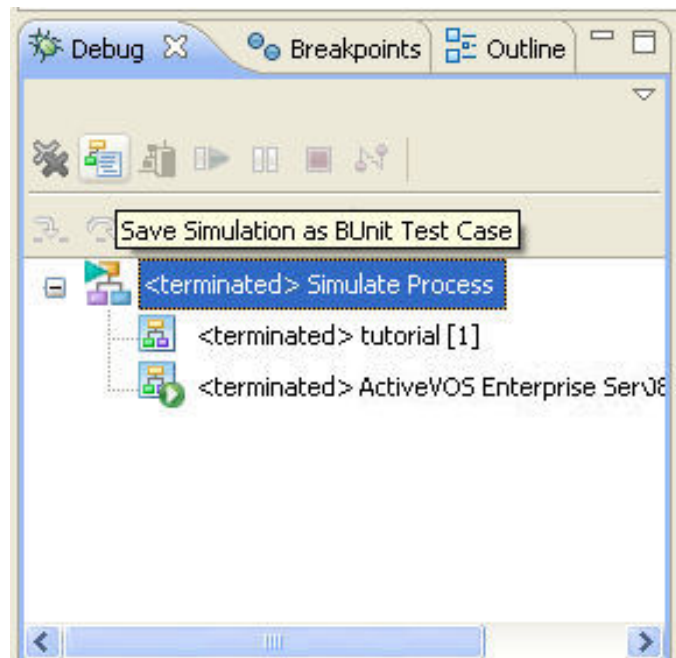
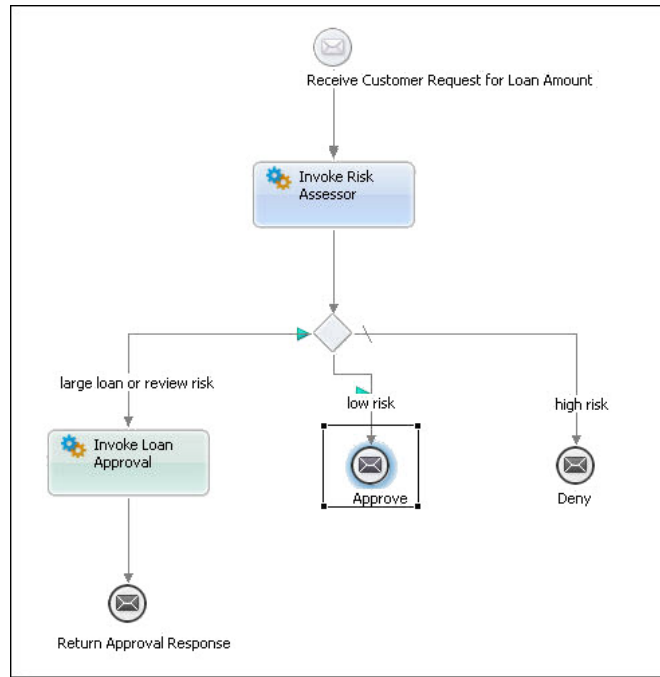


#### Use the comprehensive simulation, unit testing, and remote debugging capabilities

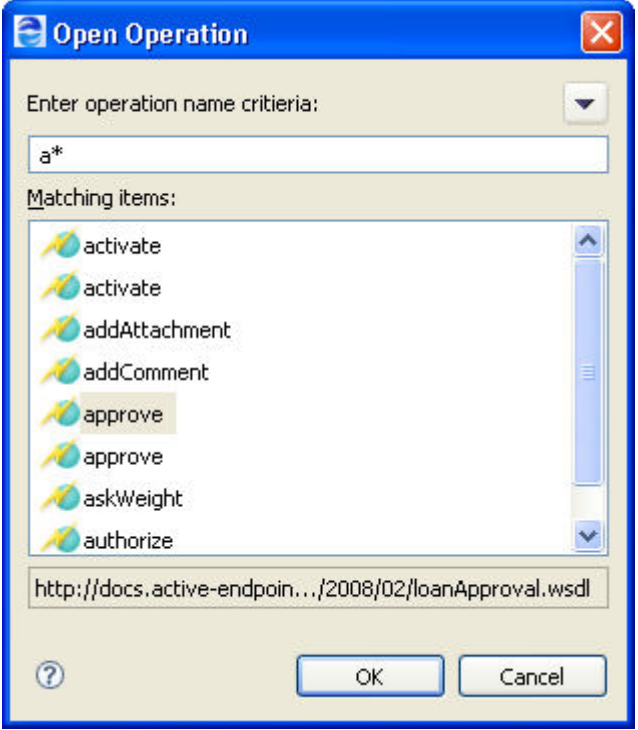
Simulate execution of a process by generating sample data values for the messages and stepping through various execution paths. Process Developer's internal execution engine provides runtime behavior for testing purposes.



Save a simulation as a BUnit test. Rerun all or only failed BUnit tests or suites to verify normal process execution.



Easily find the declarations, types and data files in your workspace by using the various Open dialogs on the Navigate menu.

	
<p><b>Starting the Process Server from within Process Developer (On-Premises Only)</b></p> <p>For remotely debugging your processes, start the server engine, and complete the deployment steps. View the Process Console by typing the console URL in your browser.</p>	
	<p>Start the server:</p> <ul style="list-style-type: none"> <li>- Select the Servers view in the lower right of the workspace.</li> <li>- Right-click and select New &gt; Server. Select the Process Server and click Finish.</li> <li>- Select the Start the Server button. As the server starts up, you see start up tasks scroll in the Process Console. Files are deployed to the embedded server each time you start it.</li> </ul> <p>View the Process Console by selecting the Process Console toolbar button. A browser opens with the following URL:</p> <p><code>http://localhost:8080/activevos</code></p> <p>Change to a different port if this one is already in use on your computer.</p>

## BPMN Design Hints

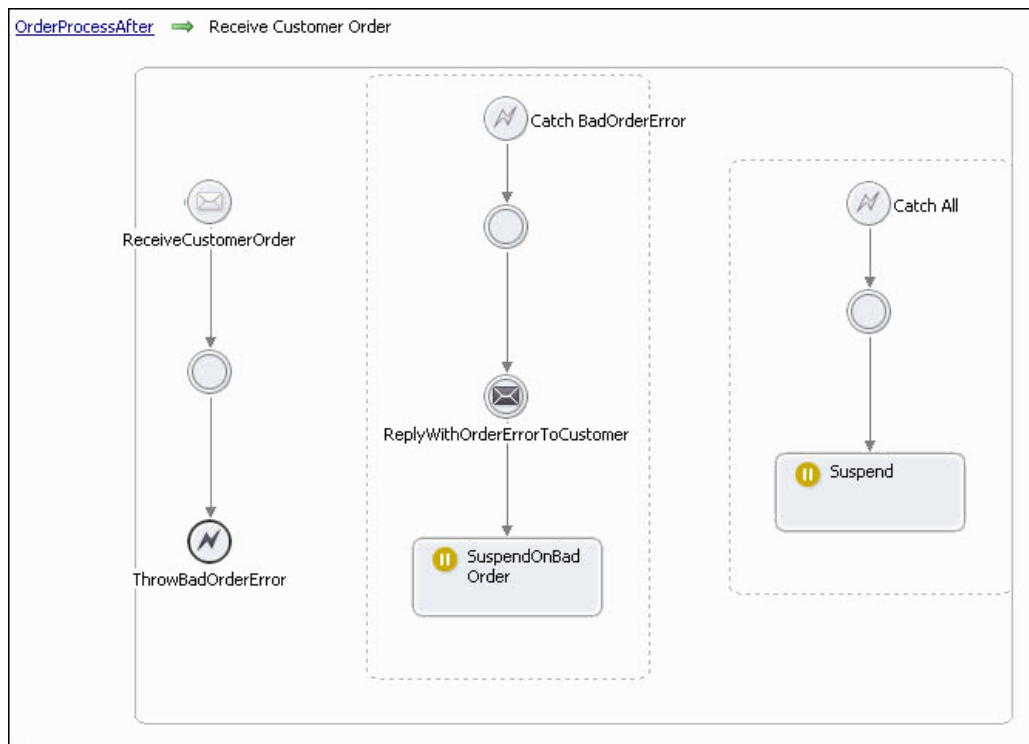
Business Process Model and Notation (BPMN) is a graph-oriented notation standard favored by business analysts and designers to model the flow of activities for analysis, documentation, and execution. Process Developer uses this standard by default for designing executable BPEL processes.

### Organize High Level and Detailed Views of a Process

Organize your process into units of work, represented by scopes. A scope is a container for any number of activities and control flows. Collapse the scopes, using a right-mouse menu item. Collapsed scopes create a high-level view of a process, as the example shows.



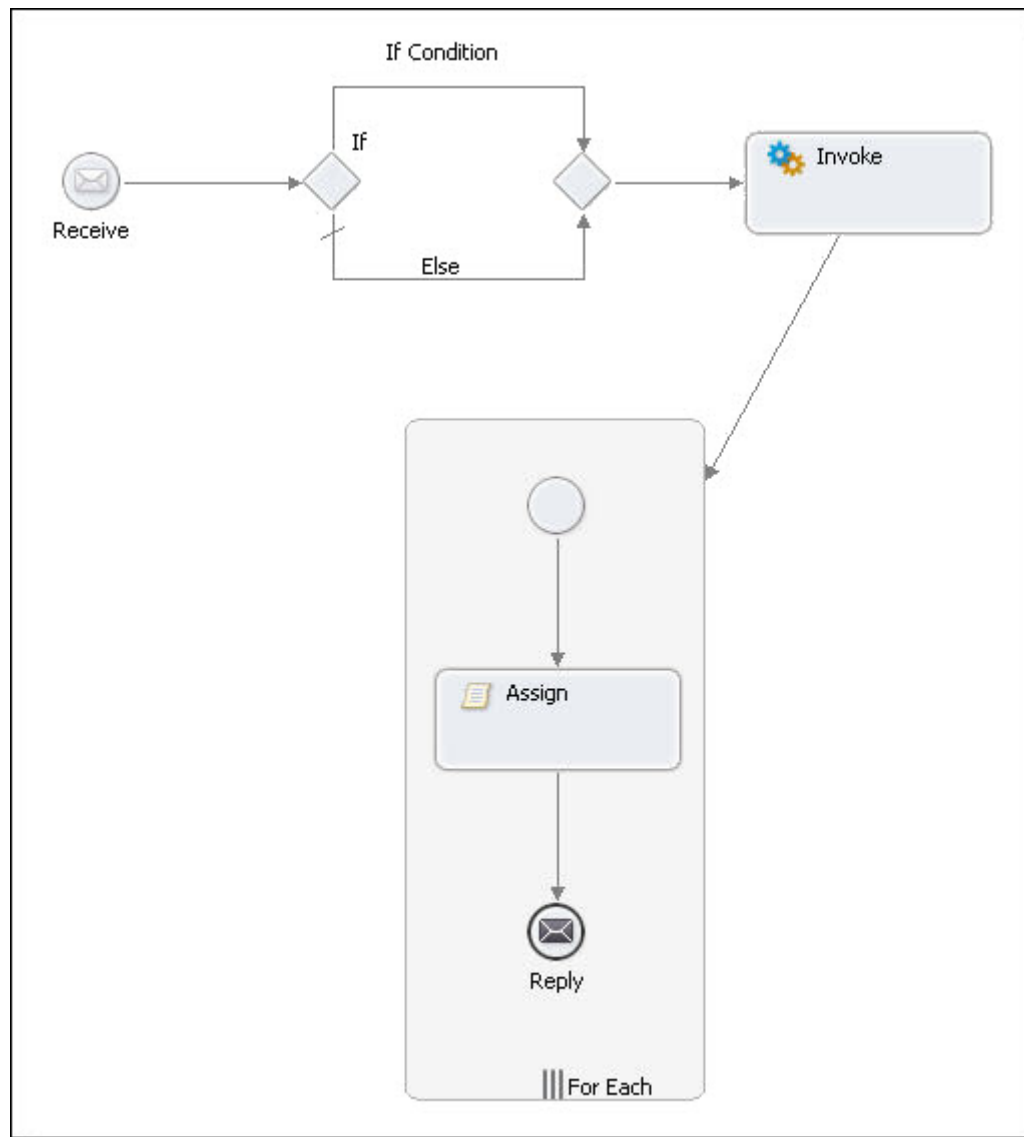
Select the plus sign (+) or the right-mouse menu option, Go Into Activity, to view the activities contained in the scope, as shown below. Select the navigation trail at the top of the window to return to the full view of the process.



### Select Horizontal or Vertical Layout

Select the layout suitable for your process and individual activities in the process.

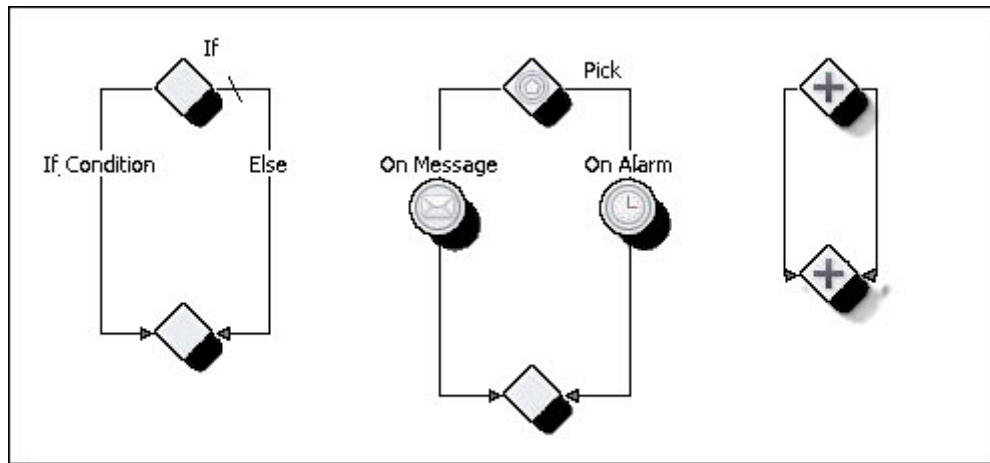
If activities are contained, for example inside a sequence, the parent container sets the layout. If activities are connected with links, you can select a different layout for each connected activity. In the example below, the horizontal sequence containing three activities is linked to a vertically aligned activity.



### Use Structured Control Flows

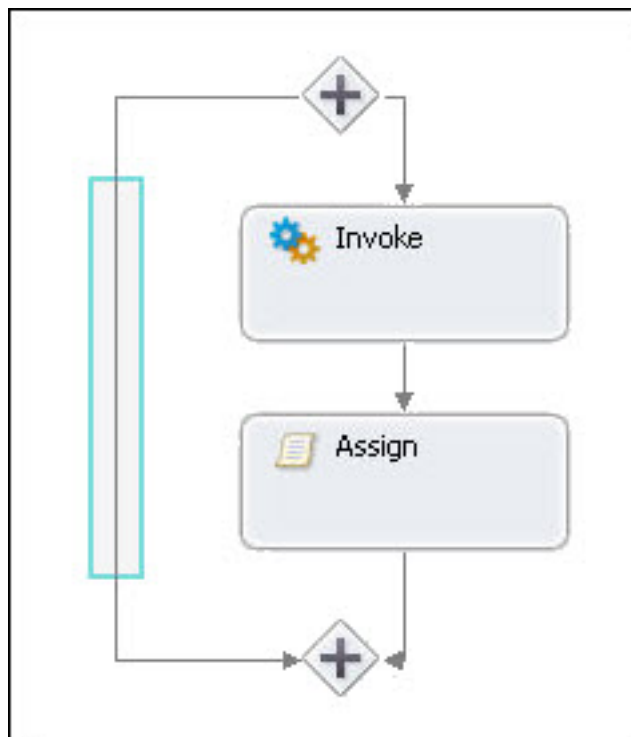
Take advantage of the structure built into activities.

The if, pick, and fork join activities are each pre-built with two paths, as shown in the example. You can add more paths or delete existing paths as desired. For example, execute several activities in parallel by adding multiple paths to the fork join.



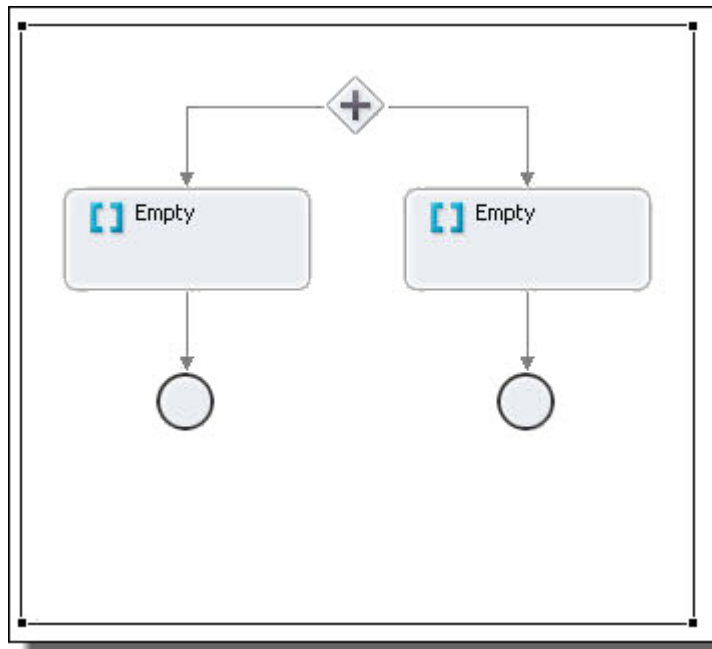
### Add Activities to a Path of If, Pick, Fork Join

Drag any activity from the palette to the implicit sequence, as shown by the blue border in the fork join example. Each new activity you add to a path is in sequence, as shown on the right-hand path.



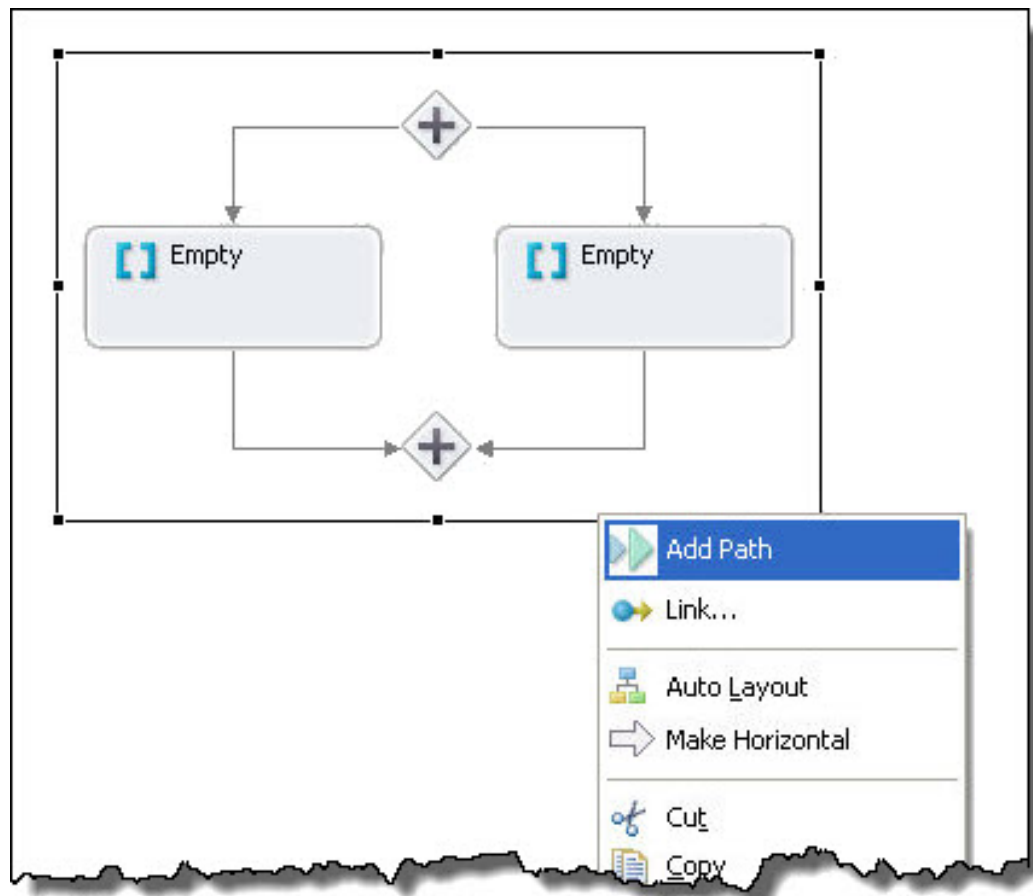
### Add End Events to Paths of If, Pick, Fork Join

If no activities follow a fork join, if, or pick, you can remove the merge diamond by adding end events, as shown in the fork join example.



#### **Add and Delete a Path of If, Pick, Fork Join**

Use the right-mouse menu to add or delete a path, as shown in the fork join example.



### Double-click to Add Expressions and Conditions

Quickly add a condition to an if, while, or repeat until by double-clicking on the activity label. The appropriate builder opens. Double-click also opens an assign activity and the transition builder for a link.

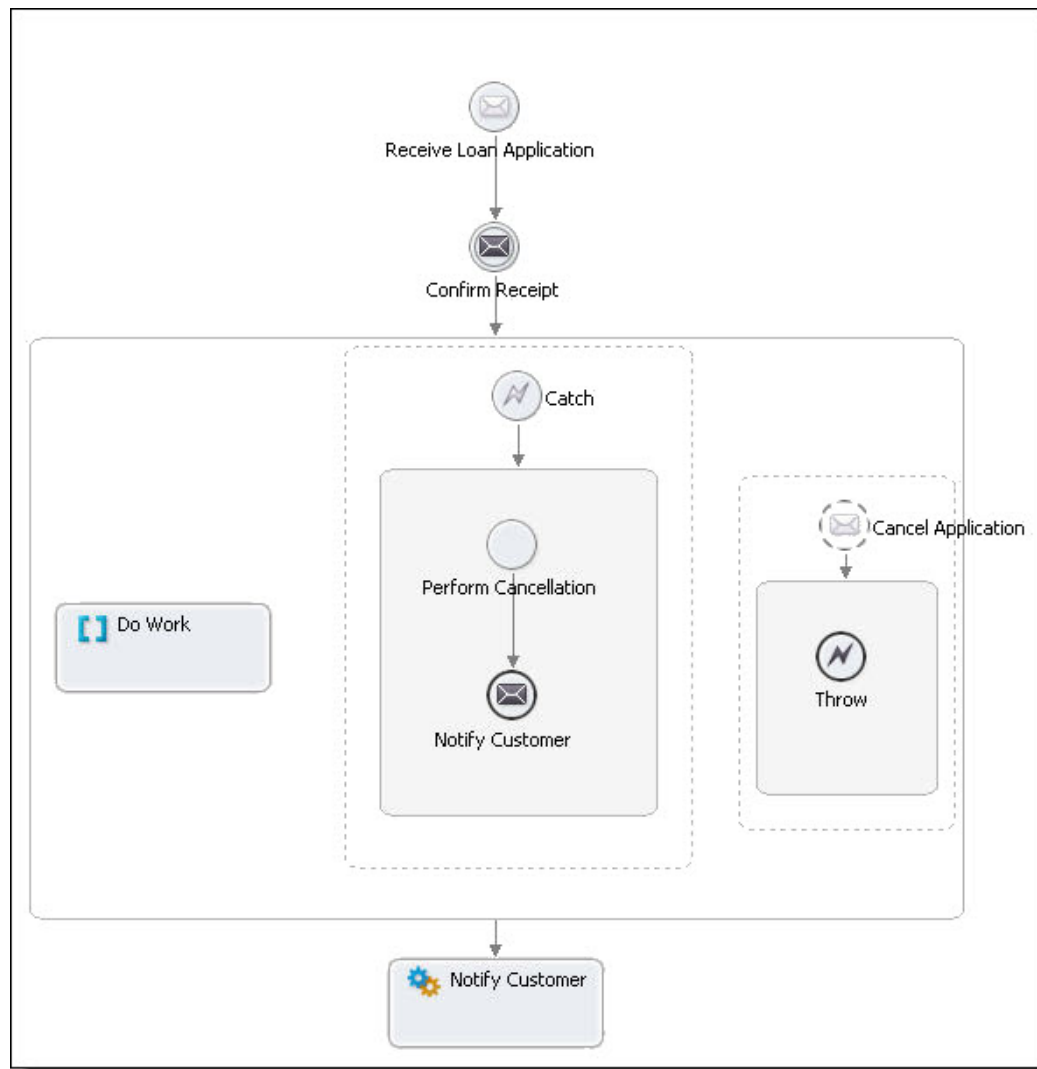
### Use Boundary Events for a Streamline View of Event Handling

You can create an uncluttered view of process events by using boundary events instead of normal scope handlers.

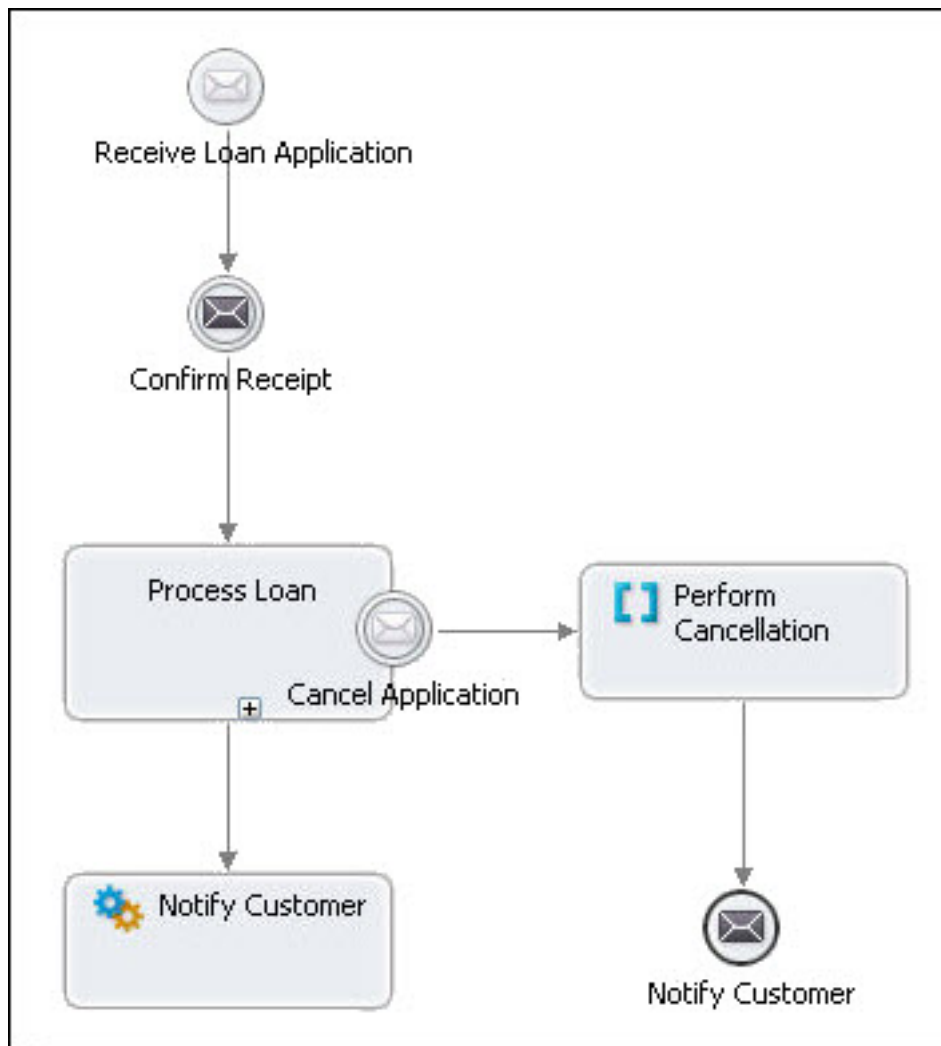
In the following example, you see in the top picture a scope with a catch and an onEvent handler. The onEvent throws a fault to the catch to cancel a loan application. In the bottom picture, the same events occur by using an interrupting onEvent which links to the cancellation work.

The bottom picture is easier to understand. Note that interrupting onEvent and onAlarm events are Informatica Business Process Manager extensions to WS-BPEL 2.0

Figure 1: Scope with normal catch and event handlers:



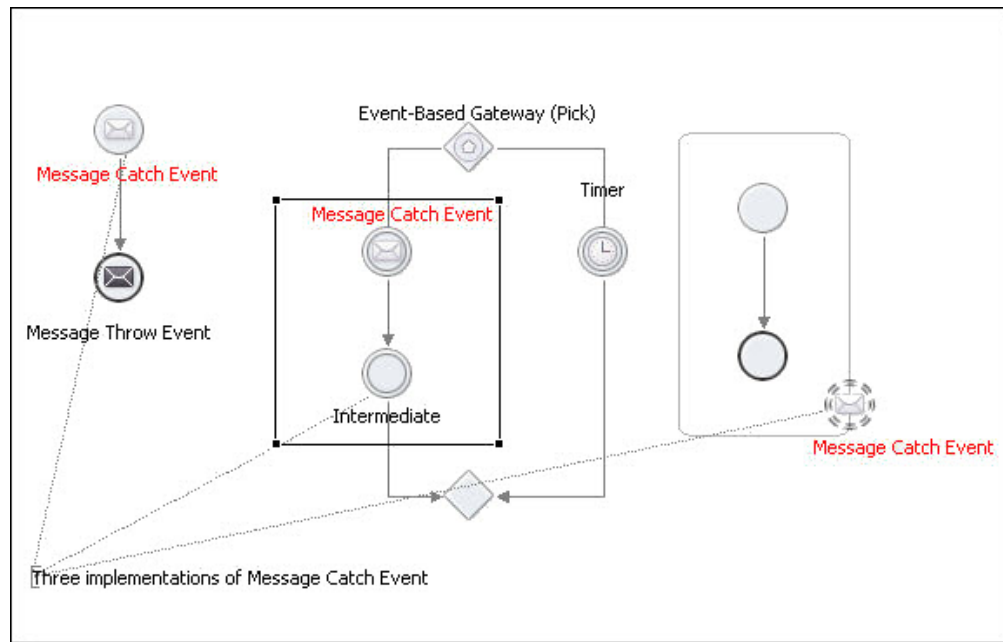




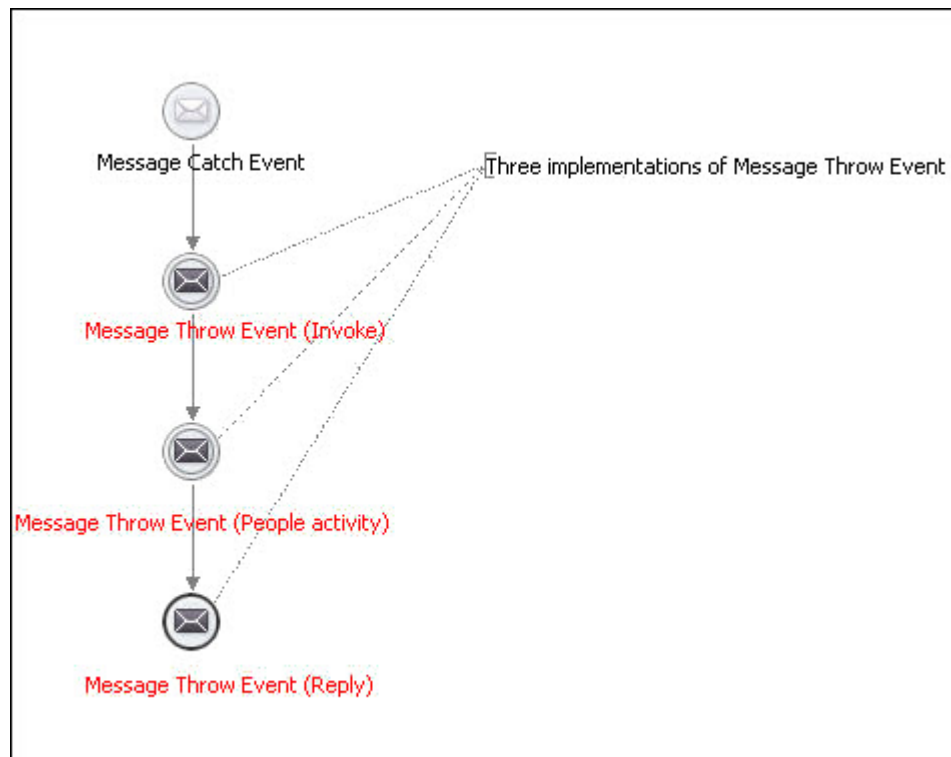
### More examples

Here are some notes on BPMN Usage

- A none intermediate event is traditionally used to note a milestone in the process. For example, the quote request is now in progress. At such a milestone, it is common that someone would want to be able to assign to a status variable. Thus an intermediate none event labeled in progress would, under the covers, assign the string in progress to the status variable. Alternately, a none intermediate event can be implemented as an empty.
- Drag a message catch event to a boundary of a bordered activity and it becomes an OnEvent event handler. Drag it to an event-based gateway and it becomes an OnMessage. Drag it anywhere else and it becomes a receive. A message catch event is a waiting event, waiting for something to trigger it.



- Drag a message throw event to the canvas and select the implementation desired: invoke, people, or reply. A message throw event is an outbound event, generating an event.



# Glossary

**.bpel.** BPEL file extension.

**.bpr.** Business process archive file extension.

**.bprd.** Business process archive descriptor script file extension.

**.bunit.** BPEL Unit testing file.

**.pdd.** Process deployment descriptor file extension.

**.pdef.** Partner definition file extension.

**.tfrm.** Task presentation form.

**.vbpel.** Visual BPEL file extension.

**.wsdl.** Web services description language file extension.

**.xds.** XML schema file extension.

**.xml.** XML data file extension.

**abstract process.** A business protocol description document that tells your business partners what a process will look like when it runs. It outlines the steps of process without actually filling in all the actions and data required for a running process. See Executable Process.

**activity.** A processing step; for example, a Receive activity accepts an input variable and can pass it to the next activity.

**AII.** Attribute information item. Part of an XML Infoset.

**BPEL.** Business Process Execution Language.

**BPEL Process.** A composition of partners, services and operations, and a definition of how data flows among those entities.

**BPELVariableName.** Name of a variable. Derived from XML Schema NCName.

**CII.** Character information item. Part of an XML Infoset.

**connection pool.** As a requestor application opens a connection to a data source and then releases it, the connection remains open for reuse by another requestor. In this way, a pool of open connections is built, with complete authentication information and connection properties. If available, the connection requests to the same data source are satisfied from the pool rather than by making a hard connection on demand, which enhances performance of the requests.

**container.** A structured activity that sets rules and conditions for executing child activities; for example, a sequence organizes activities to execute in an ordered list.

**deadline or duration.** The settings for OnAlarm events in a pick activity, event handler, or task.

**DII.** Document information item. Part of an XML Infoset.

**EII.** Element information item. Part of an XML Infoset.

**executable process.** A BPEL process that contains all the actual message data, operations, and partner information required for a running process. It uses the full power of data assignment and selection. See Abstract Process.

**join condition.** A Boolean expression indicating the status of a link targeting an activity.

**my role.** A child element of a partner link. A partner link defines "my role" as the role played by the business process. The service called for receive, pick and event input messages is the service fulfilling my role.

**partner.** A unique collection of partner links.

**partner link.** Describes the roles that a process and service play as well as what data they can manipulate in that role. A partner link is defined by its partner link type.

**partner link type.** Describes the kind of message exchange that two WSDL services intend to carry out. A partner link type characterizes this exchange by defining the roles played by each service and by specifying the port type provided by the service to receive messages appropriate to the exchange.

**partner role.** A child element of a partner link. A partner link defines partner role as the role played by an invoked service.

**TII.** Text information item. A sequence of zero or more Character Information Items, according to document order; as such, a TII is not manifested in and of itself directly in XML serialization. When mapped to the XPath 1.0 model, it generalizes a string object (which has zero or more characters) and text node.

**URL.** A web site address, such as <http://www.informatica.com>.

## CHAPTER 2

# Introducing Business Process Execution Language

Business Process Execution Language for Web Services (BPEL) is an XML notation for defining process orchestrations based on Web services protocols. The BPEL specification is an important standard in the Web service architecture. It describes process-specific language constructs and defines how multiple Web services can be composed into coherent information systems. BPEL builds upon other standards in the Web services architecture, most notably Web Services Description Language (WSDL).

### **BPEL Resources**

The WS-BPEL 2.0 specification is available on the Informatica Web site. WS-BPEL depends on the following XML-based specifications:

- Web Services Description Language Version 1.1.
- XPath expression language
- XML Schema WS Addressing.
- See the WS-Addressing specification at the Web site of any of the specification's contributors, including IBM, BEA Systems, and Microsoft.

## BPEL for People Extension to WS-BPEL 2.0

Business Process Execution Language for Web Services (BPEL) is an XML notation for defining process orchestrations based on Web services protocols. The BPEL specification is an important standard in the Web service architecture. It describes process-specific language constructs and defines how multiple Web services can be composed into coherent information systems. BPEL builds upon other standards in the Web services architecture, most notably Web Services Description Language (WSDL).

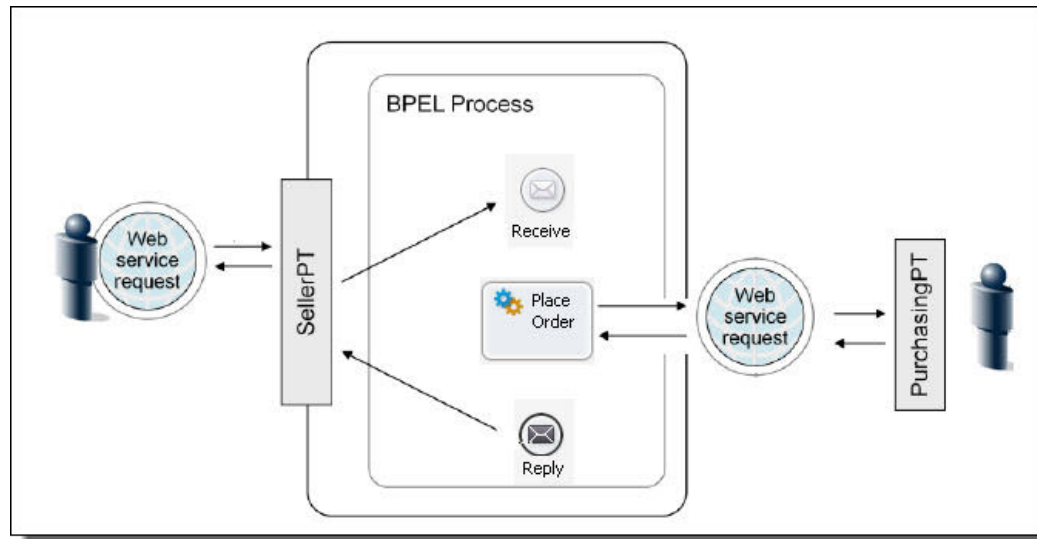
The BPEL for People plugin installs extension elements described in the Web Services Human Task (WS-HT) Version 1.0 Specification and an extension activity, called the People activity, described in the WS-BPEL Extension for People Version 1.0 Specification. The tasks, notifications, Logical People Groups, and People activity introduce human workflow into a BPEL process when a process activity requires human decision-making.

For details, see Human Tasks, which is also within this help.

# What is a BPEL Process

A BPEL process is a collection of Web services whose interactions are choreographed in a defined manner. Each service is a participant that performs some type of processing. Services can be highly granular (for example, calculate a rate) or very large in scope (for example, process an order).

In the following figure, the process is the Seller participant, accepting a purchase order from a customer through a Web service input message. The Seller then places the order with the Purchasing participant. To end the process, the process returns an acknowledgement to the customer if the order was fulfilled.

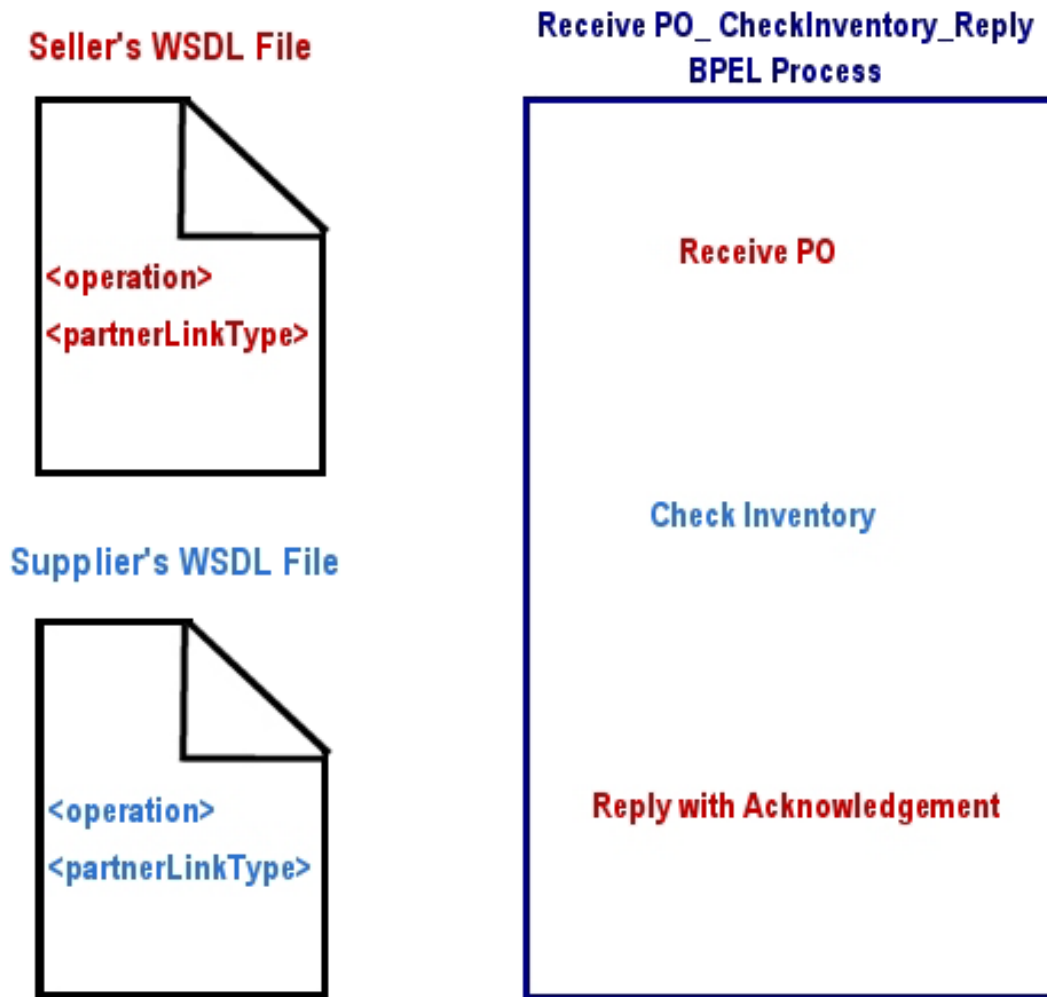


After participants send the acknowledgement, they could send other messages to the customer, such as shipping notices and invoices.

Because you want to have this type of interaction with the customer for many purchase orders, as well as with other customers, you build one business process, and it acts as a template for business process instances.

The business process you build has a setting to "create an instance" so that each time a new purchase order arrives, a new process is created. The new process handles all the related interactions for it, keeping interactions separated for each purchase order for each buyer.

The BPEL process definition uses as input the definitions from Web Services Description Language (WSDL) files. These files contain interface information that can be shared with the outside world. A process developer selects information such as partner link types and operations to define the process steps, as shown in the following illustration.



A BPEL process coordinates these interactions and composes them into a straight-through or long-running flow. For example, if an exception occurs while a process is executing, activities could be reversed or undone, and a BPEL process provides the techniques for correlation, compensation, and fault and event handling.

## BPEL Process Definition Elements

The main sections of a Business Process Execution Language (BPEL) definition are shown in the following code sample.

```
<process>
  <!-- Definition of roles of process participants -->
  <partnerLinks> ... </partnerLinks>
  <!-- Data and state variables used within the process -->
  <variables> ... </variables>
  <!-- Correlation comment -->
  <correlationSets> ... </correlationSets>
  <!-- Exception management -->
  <faultHandlers> ... </faultHandlers>
  <!-- Message and timeout event handler -->
  <eventHandlers> ... </eventHandlers>
  <!-- Processing steps -->
```

```
    activities*  
</process>
```

## Informatica Extensions to WS-BPEL 2.0

In addition to fully supporting the WS-BPEL 2.0 specification, Informatica Business Process Manager has the BPEL extensions described in this topic.

### Process Level Extensions

- Compensation handler and termination handler  
For details, see "Process Element and Properties" in this help.

The extension namespace is [http://www.activebpel.org/2006/09/bpel/extension/process\\_coordination](http://www.activebpel.org/2006/09/bpel/extension/process_coordination).

- Query handling including Create XPath and Disable Selection Failure  
For details, see "Process Element and Properties" in this help.

The extension namespace is [http://www.activebpel.org/2006/09/bpel/extension/query\\_handling](http://www.activebpel.org/2006/09/bpel/extension/query_handling).

### Activities

The extension activities are "Suspend", "Break", and "Continue", which are described elsewhere in this help.

The extension namespace is <http://www.activebpel.org/2006/09/bpel/extension/activity>.

### Links

Links are used in WS-BPEL as synchronization constructs where one activity can be either a target or a source of another activity, but not both at the same time. The link extension allows loop-back functionality. See "Extension for Links" elsewhere in this help.

The extension namespace is <http://www.activebpel.org/2009/06/bpel/extension/links>.

### Custom XPath Functions

There are several custom functions, as described in "Using the Expression Builder" in this help.

The extension prefix `abx` is for the namespace <http://www.activebpel.org/2006/09/bpel/extension>.

### Implicit Scopes Variables

To allow for a streamlined visual display of a process, Process Server can eliminate assign activities and process variables used to map data. Instead, you can add data mapping within receives, replies, invokes, and people activities. When you map data within these activities, internally scoped variables are added to contain the data. See "Input Variable" and "Output Variable" elsewhere in this help.

The extension namespace is <http://www.activebpel.org/2009/02/bpel/extension/ignorable>.

### Interrupting Boundary Events in BPMN

You can treat event handlers as BPMN boundary events and set them to terminate the main activity. See "Adding Boundary Events" elsewhere in this help.

The extension namespace is <http://www.activebpel.org/2009/02/bpel/extension/ignorable>.

### Undeclared SOAP Fault (Java Name)

Informatica Business Process Manager includes custom functions to catch undeclared faults and allows you to catch a fault by Java name. For details, see "Catching Undeclared and SOAP Faults" elsewhere in this help.



The required extension namespace is `http://www.active-endpoints.com/2004/06/bpel/extensions/`.  
Process Server also uses this namespace to catch system errors.

## CHAPTER 3

# ActiveVOS Tutorial

The ActiveVOS tutorial guides you through creating a Business Process Execution Language (BPEL) process, testing the process in simulation mode, then deploying it to the server (and using remote debugging if you deploy on-premises).

You will create a loan approval process definition. This definition describes the interactions among three participants, as well as the control flow and decisions based on exchanged data. It is based on Web Services Description Language (WSDL) documents that describe the messages and business operations used in the process.

After you finish this tutorial, you will have worked with many of ActiveVOS capabilities.

The tutorial is divided into the following sections:

- **Part 1: Starting a New Process**  
You will learn how to create the tutorial project, create and open a process document, and become familiar with the ActiveVOS environment.
- **Part 2: Planning and Designing a Process**  
You will learn how to use the Process Editor and its graphical toolkit to develop the sample loan approval process BPEL definition.
- **Part 3: Creating a Process Service Consumer Participant**  
You will learn to add a participant role for the process to provide the BPEL-required partner link.
- **Part 4: Creating Partner Service Provider Activities**  
You will learn the powerful shortcuts in Process Developer for creating receiving, replying, and invoking activities in a process by starting with a WSDL operation.
- **Part 5: Adding Process Activities and Properties**  
You will learn how to:
  - Complete a process definition by adding control flows.
  - View your process in both a graphical and hierarchical view.
  - Create namespaces, variables, and activities manually and define properties for them.
- **Part 6: Adding Fault Handling**  
You will learn how to catch errors sent to the process.
- **Part 7: Adding Compensation and Correlation**  
You will read a short discussion of these topics, which are not included in the sample process.
- **Part 8: Simulating the Process**  
You will learn how to simulate process execution with sample data values.
- **Part 9: Deploying the Process**  
You will learn how to create a process deployment descriptor file and package several files into a business process archive for deployment to the Process Server.
- **Part 10: Creating a Form to Run the Process**

You will learn how to deploy, run, manage, and test a process on the Process Server. You will create a form to start the process from Process Central.

- **Part 11: Debugging Your Process Remotely**

You will learn how to set up remote process launch configurations to debug a process running on the server.

- **Part 12: Using the Web Services Explorer to Start a Process**

You will learn how to start your process from an Process Server, such as JBoss or WebLogic by using the Web Services Explorer.

## Part 1: Starting a New Process

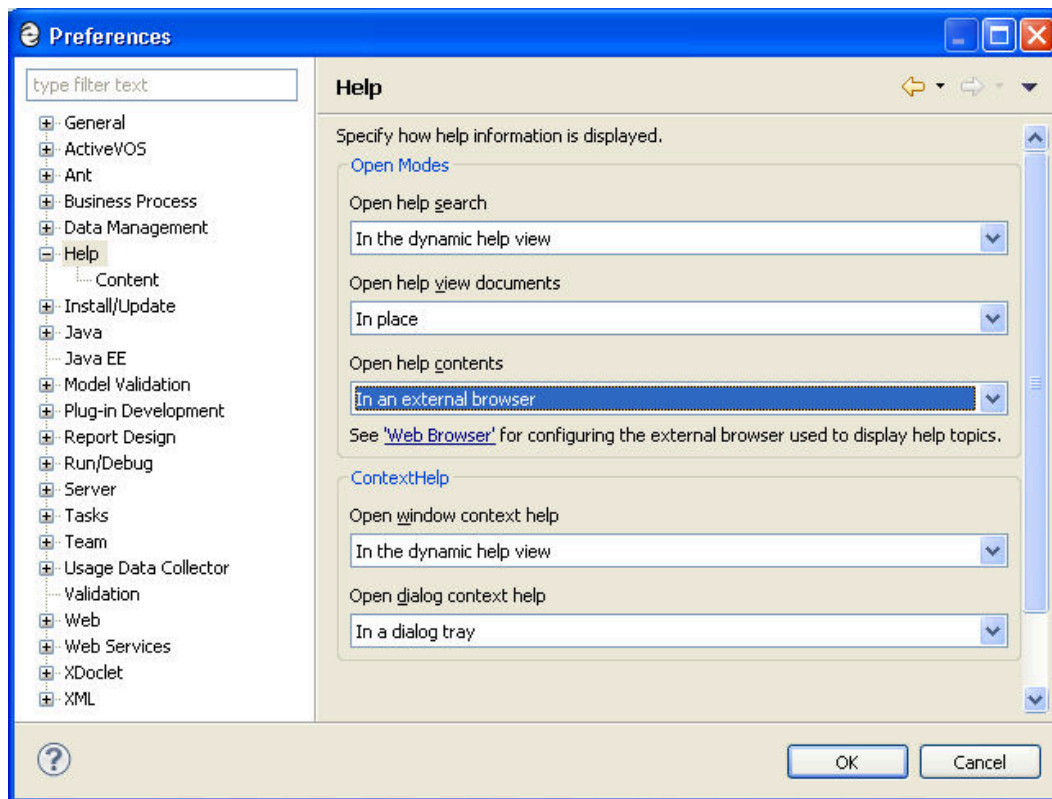
You can use Process Developer to design a process using top-down or bottom-up techniques. In the top-down technique, you sketch out your process by dropping down activities within a bounding box that automatically contains sequenced activities. You then add all the necessary information to bind the activities to an implementation and add decision-making to make your sketch a working process. In the bottom-up technique, you use the definitions of the implementation that are available when you begin process creation. This tutorial covers both design techniques.

After completing Part 1 of the tutorial, you will be able to:

- Create a new orchestration project.
- Understand the nature of an orchestration project.
- Create a new process document in the Tutorial orchestration project.
- Familiarize yourself with the Process Developer tools that help you create a valid process.

When you are done, go to [“Part 2: Planning and Designing a Process” on page 41](#) .

**Tip:** Set a Help Preference to display Help in an external browser. This lets you see this tutorial at all times, especially when a modal dialog is open. To set a Help Preference, select **Window > Preferences > Help**. Next, select *In an external browser* within the "Open help contents" picklist, as shown in the following illustration.



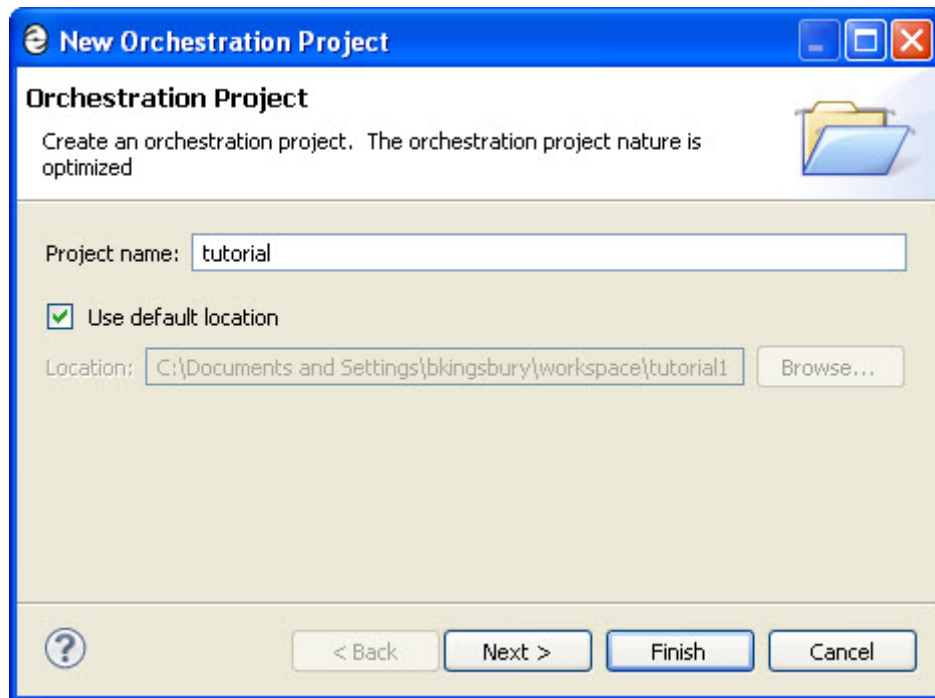
**Note:** The illustrations of dialog boxes and wizards (for example, font and button color) that you will see in this tutorial can look different from one version of Eclipse to another and your operating system may also display them differently than are shown here.

## Step 1: Create the Tutorial Orchestration Project

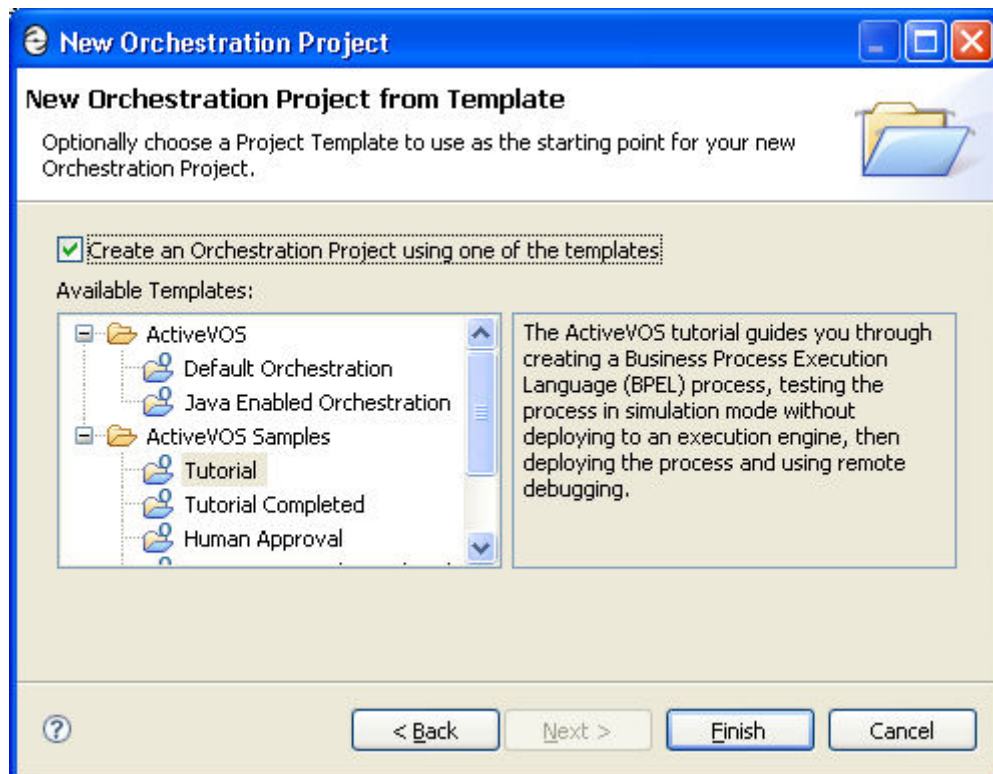
You will create an Orchestration Project in this tutorial. An "orchestration project" is an Eclipse project with a special nature for including the resources required for composing BPEL processes. These resources include WSDL, schema, deployment files, and other specialized resources like Service References. In addition to folders, an Orchestration Project has builders, which ensure that all orchestration files are valid, helping you to complete error-free orchestrations.

Like all Eclipse projects, an orchestration project is a container to store files, and it resides in your default location, the Workspace folder selected when Process Developer was installed. Orchestration projects are also created in the same Workspace location in the file system.

1. Select **File > New > Orchestration Project**.
2. Type in `Tutorial` for a project name, as shown.

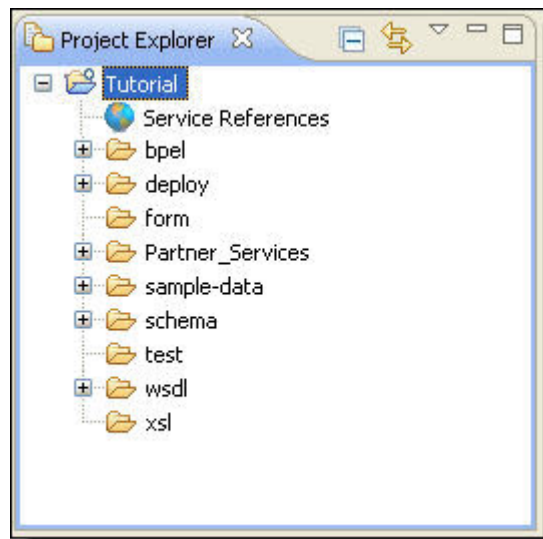


3. Click **Next** and select the Tutorial template.



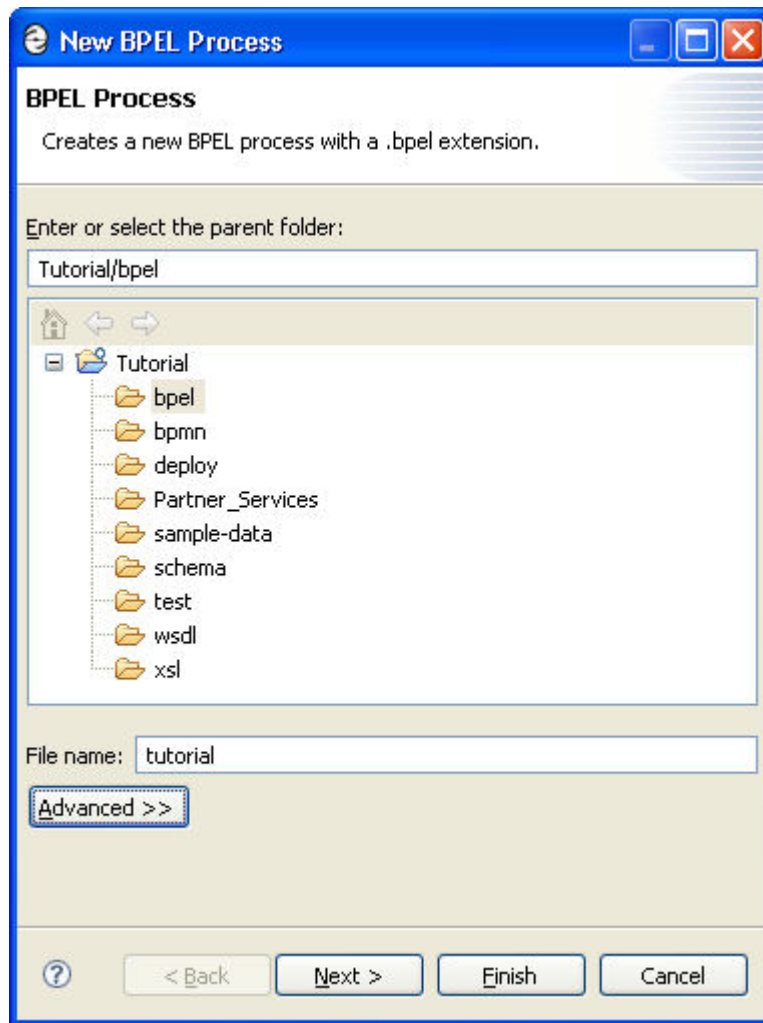
4. Click **Finish**.

Your Project Explorer view should look like the following illustration. Notice that Process Developer created folders to contain the resources needed for a BPEL-based orchestration. As you proceed through the tutorial, you will be using each of these folders in the Tutorial project.



## Step 2: Create a New Process File in the Tutorial/bpel Folder

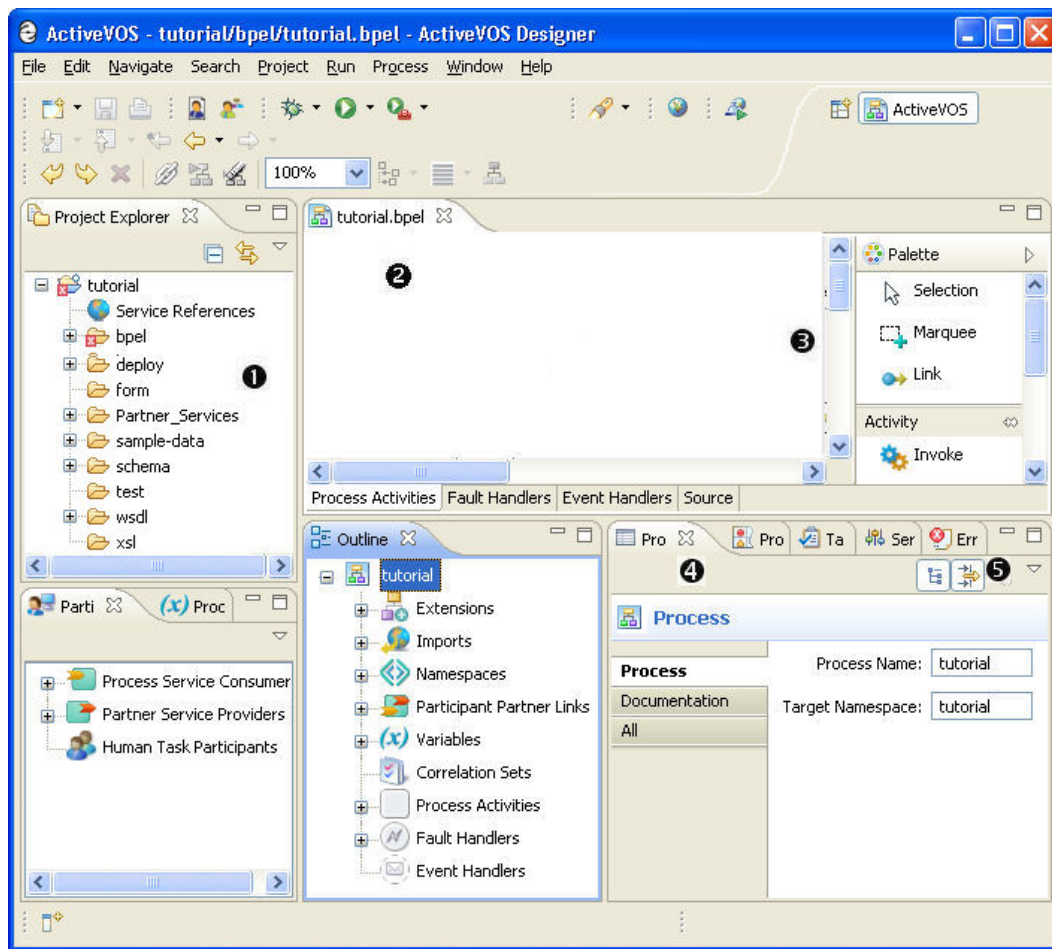
1. Select **File > New > BPEL Process**.
2. Select the `bpel` folder.
3. In the file name field, type in `tutorial` for your BPEL file. The `.bpel` extension is automatically added.



Notice the **Advanced** button. Pressing this button displays settings that override defaults that you can set in Preferences. It won't be used in this Tutorial, but you may want to press it to see what's there.

4. Click **Finish**.

Your new file opens as a blank document in the Process Editor, as shown in the following illustration.



Notice the following user interface features:

1	Process file are listed in the Project Explorer view. Double-click on a Project Explorer file to open it.
2	A newly created process file is opened automatically in the Process Editor. The Process Editor canvas is blank to begin with. When you open multiple files, they are displayed as different tabs. To switch between open files, click on a tab.
3	To create a process, you will use the palette. Each palette group contains a set of icons, such as the Task group. You can select a palette icon and drop it onto the canvas. The palette is closed by default. Rest your mouse on the palette bar to auto-open it, or select the <b>Show Palette</b> arrow to open it.
4	The Properties view displays the attributes for the object in focus. In the illustration above, the <code>tutorial.bpel</code> process is in focus in the Process Editor. If a file is selected in the Project Explorer, a different set of attributes is displayed.
5	The tabs along the top of a view indicate that several views are stacked together. Select a tab to display a view. Tip: You can close (hide), minimize, maximize, move, and rearrange views as desired.

If you select the Problems tab, you may see two error messages. One says, "No activity designated to create instance" and "Container /process is missing a required activity." These messages are part of BPEL validation. You can ignore them for now.

**Note:** The Tutorial orchestration project includes a Cheat Sheet. A default project does not. The Cheat Sheet view is usually closed.



## Part 2: Planning and Designing a Process

[“Part 1: Starting a New Process” on page 35](#) covered creating a new project and a new BPEL file.

After completing Part 2 of the tutorial, you will be able to:

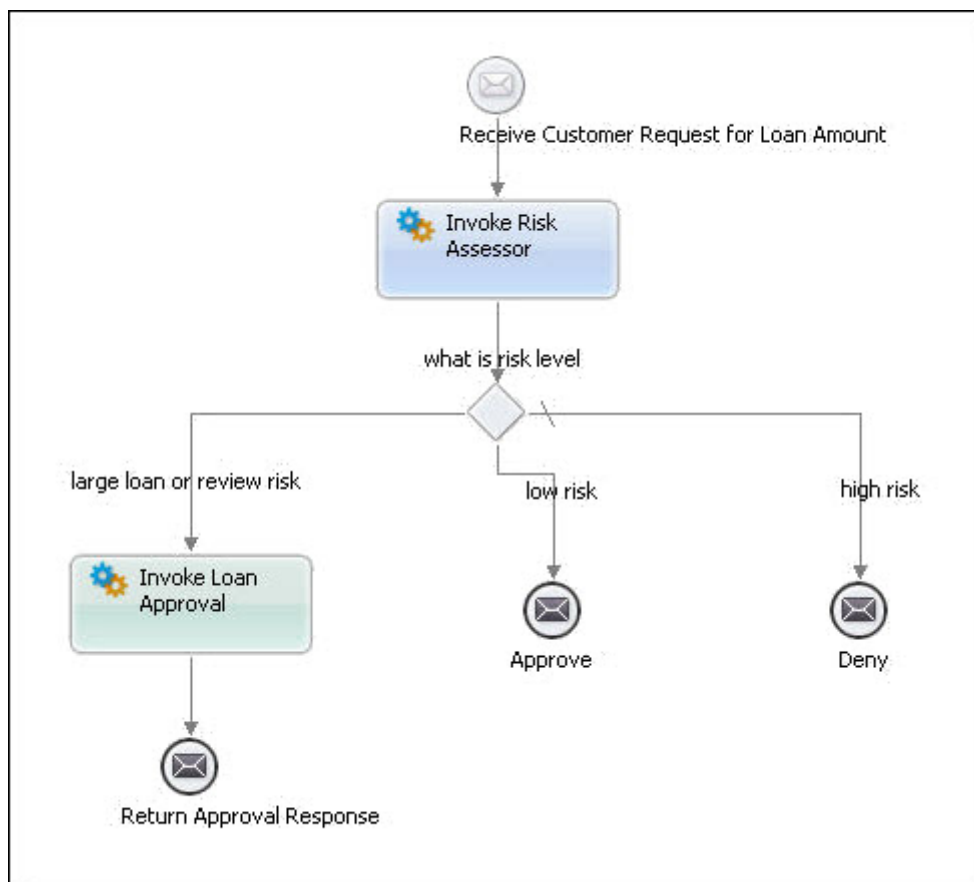
- Understand the design of the loan approval process.
- Understand top-down vs. bottom-up design.
- Use layout features of the Process Editor to optimize process display.
- Review BPEL validation messages.

At the end of this tutorial part, you will understand the loan approval process we will build.

### What is the Loan Approval Process?

The Loan Approval process starts by receiving a customer request for a loan amount. The risk assessment Web service is invoked to assess the request. If the loan is small and the customer is low risk, the loan is approved. If the customer is high risk, the loan is denied. If the customer needs further review or the loan amount is for \$10,000 or more, the request is sent to the approver Web service. The customer receives feedback from the assessor or approver.

A top-down design would begin by adding activities to the Process Editor without adding any valid attributes to them; for example:

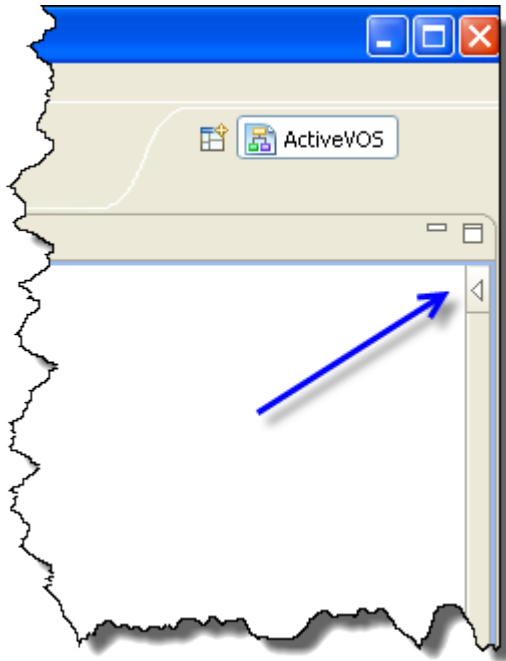


This part of the tutorial familiarizes you with the Process Editor, demonstrating features you can use to create a top-down design. You will not create a complete top-down design.

## Step 1: Create a Receive activity

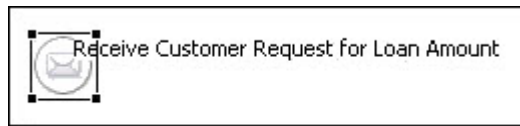
A BPEL process begins by receiving an input message. This means a top-down design would start with either a Receive or a Pick activity, since they are structured to receive data from a business partner's Web service.

1. If the `tutorial.bpel` file that you created in Part 1 is not open, open it in the Process Editor.
2. If Cheat Sheets view is open, close it to make more room.
3. Click the **Show Palette** arrow.



4. From the **Catch Event** palette, drag a Message icon to the canvas. The activity is labeled with Message.
5. Make sure that the message is selected. A selection box encloses it. When it is selected, the Properties view displays Receive attributes.
6. To make the activity name more meaningful, do one of the following:
  - Click on the activity label Message and type in `Receive Customer Request for Loan Amount`.
  - In Properties view, type the following in the Activity Name field, `Receive Customer Request for Loan Amount`.

Here's what your activity should look like.

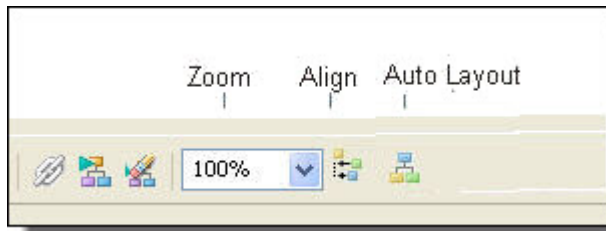


## Step 2: Working with Layout Features in the Process Editor

You can use several features in the Process Editor to optimize the display of your process. You will now complete a short exercise to demonstrate its layout features.

1. From the **Task** palette, drag an Abstract task to the canvas.

2. Try the following features by selecting them from the toolbar:



- Use Auto Layout to layout activities in a grid
  - Select two or more activities and Align them
  - Magnify the display area with Zoom
  - Display the Thumbnail view to select one area of your design to view. By default the Thumbnail view is closed. Open it from the Window > Show View command.
3. Delete the Abstract task.

## Step 3: Save the File and Review BPEL Validation Messages

Take a moment to save your file. You will notice a change to your Receive activity icon as it now has a error icon next to it. Also, this error is now within the the *Problems* view. These "errors" are added to the Problems view by default when you save your file. They relate to the validation of BPEL code generated for your process. You can see the BPEL XML code in the Source view of the Process Editor.

The errors listed in Problems view will automatically disappear as you complete this BPEL process in the tutorial.

Continue to ["Part 3: Creating a Process Service Consumer Participant" on page 43](#) .

## Part 3: Creating a Process Service Consumer Participant

If you have not already done so, complete or review ["Part 1: Starting a New Process" on page 35](#) and ["Part 2: Planning and Designing a Process" on page 41](#) .

In Part 2 of the tutorial, you learned how to create a top-down design by arranging BPEL activities on the Process Editor canvas. Now you will begin creating an executable process based on WSDL files.

In the Project Explorer view of Process Developer, you should have the following:

- Tutorial project
- `tutorial.bpel` that you created in Part 1

After completing Part 3 of the tutorial, you will be able to:

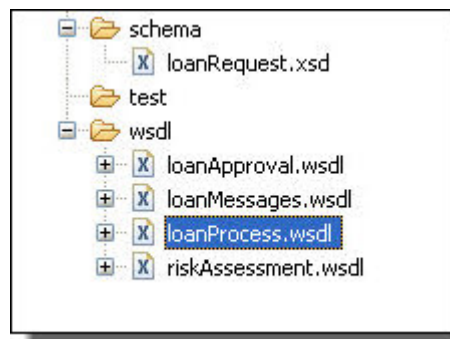
- Understand the imported WSDL and schema for the loan approval process and orchestrated services.
- Use the Participants View to create a Process Service Consumer.
- Create the receive and reply activities.

## Step 1: Viewing the WSDL and Schema

A Web Services Description Language (WSDL) file describes business operations that are invoked to carry out the activities of a BPEL process. WSDL files, and the schema types and elements they import, are required in order to create a valid executable BPEL file. A good practice is to include your business partners' WSDL and schema files in your orchestration project so that you can easily move the files in the project to a deployment package.

If you do not have access to WSDL files, you can create them in Process Developer.

Normally you would create an orchestration project folder and import (or create) WSDL files into it. To save time, the WSDL files are already imported, as shown in the following illustration:



The WSDL and schema file definitions are as follows:

loanRequest.xsd	Schema that defines the data types used in messages. The schema is imported into loanMessages.wsdl.
loanApproval.wsdl	WSDL describing the approve operation for the loan approver partner service. The service is invoked to approve or deny requests for large loans.
loanMessages.wsdl	WSDL containing the message definitions for data exchanged among the service partners. This WSDL is imported by the other WSDLs.
loanProcess.wsdl	WSDL for the participation of the BPEL process itself, describing the request operation. The BPEL process starts by receiving a loan applicant's request for a loan.
riskAssessment.wsdl	WSDL describing the check operation for the risk assessment partner service. The service is invoked to check the risk level of an applicant applying for a small loan.

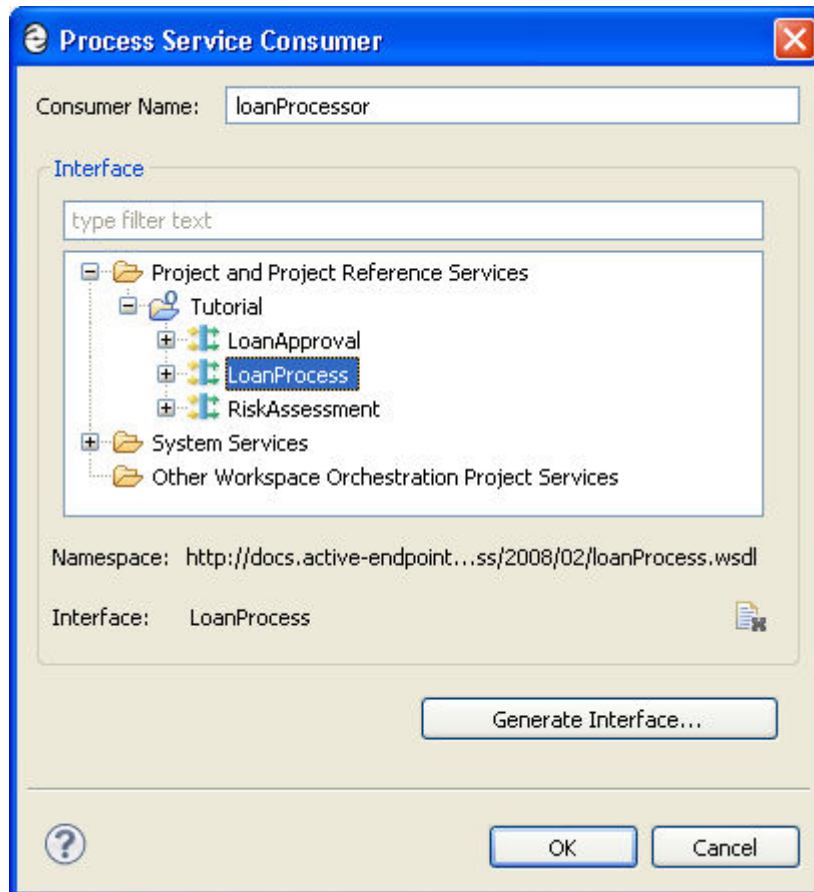
## Step 2: Using the Participants View to Create a Process Service Consumer

Participants are the Web services and clients using your process and the partner Web services your process uses. A participant exposes an interface to exchange messages with other services.

The loan approval process has three participants: one participant is the process itself, playing the role of receiving a customer request and replying to the customer. The other two participants are partner services that are created later.

1. Right-mouse click on Process Service Consumers and select New Process Service Consumer. Each process plays at least one participant role, that of being consumed by a Web Service to start the process.

2. In the Interface tree, expand Project and Project Reference Services to see three port types listed in the Tutorial project. A port type is a required element of WSDL. It describes the operation to use for invoking a Web service and the messages to be exchanged between Web services. These port types are defined in the WSDLs that are provided with the Tutorial orchestration project. Select the LoanProcess port type.
3. The default Consumer Name is Process\_Consumer. Rename this to `loanProcessor` to identify the participant role that this process is playing.

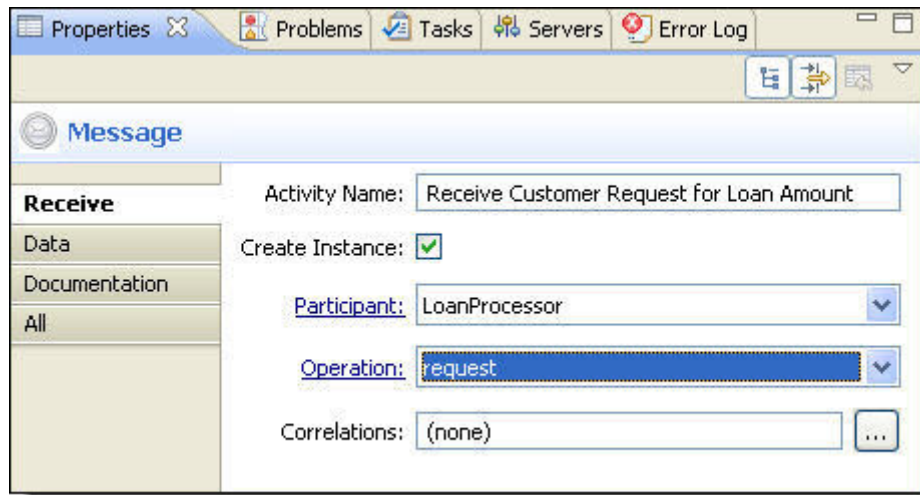


4. Click **OK**. The new participant is displayed, and you can expand it to show the interface details from `loanProcess.wsdl`.

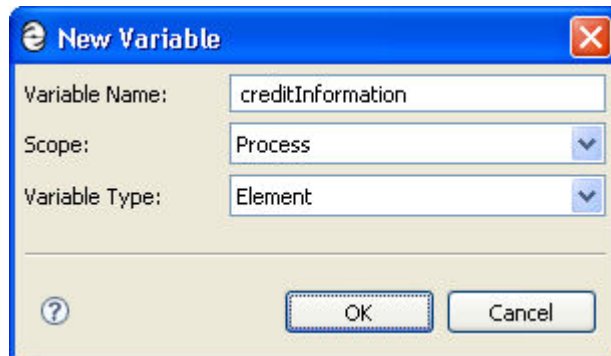
### Step 3: Complete the Interface Details for the Receive Activity

The `loanProcessor` participant is consumed by the request operation to begin the process. Implement this by filling in the details for the Receive activity.

1. On the Process Editor canvas, select the Receive activity.
2. In the Properties view, select the Receive tab on the left, and then select the drop-down arrow in the Participant field, and select `loanProcessor`.
3. In the Operation field, select the request operation, as shown.



4. Select the Data tab on the left.
5. From the Assignment Type picklist arrow, select Single Variable.
6. From the Variable picklist, select New Variable.  
The **New Variable** dialog displays a suggested name and data type, based on the schema imported into `LoanMessages.wsdl`.
7. Rename the variable to `creditInformation`, as shown.

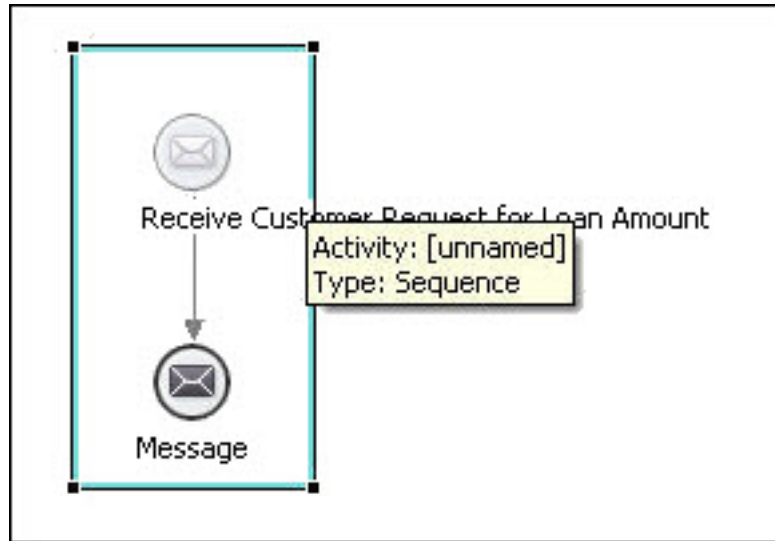


## Step 4: Create a Reply Activity

If you have not already done so, complete or review [“Part 1: Starting a New Process” on page 35](#) and [“Part 2: Planning and Designing a Process” on page 41](#) .

The request operation that the process starts with is a request-response operation. You now need to create the mandatory reply. The reply tells the customer whether or not the loan amount is approved.

1. From the Throw Event palette, drag a Message to the canvas. Place it within the bounding box surrounding the receive, as shown.



2. In the Properties view, look at the Activity Name field, then name the Reply `Return Approval Response`.
3. In the Participant field, select `loanProcessor`.
4. In the Operation field, select `request`.
5. Select the Data tab.
6. From the Assignment Type, select `Single Variable`.
7. From the Variable picklist, select `New Variable`.
8. Name the new variable `approval`.

After you save your file, you will notice that the errors that existed when you saved the file in Part 2 (see [“Step 3: Save the File and Review BPEL Validation Messages” on page 43](#)) no longer appear.

## Part 4: Creating Partner Service Provider Activities

If you have not already done so, complete or review [“Part 1: Starting a New Process” on page 35](#), [“Part 2: Planning and Designing a Process” on page 41](#), and [“Part 3: Creating a Process Service Consumer Participant” on page 43](#).

In Part 3 of the tutorial, you created a participant for the process role and the associated receive and reply activities. Now you will create the partner services that assess and approve or deny the loan request.

In the Project Explorer view of Process Developer, you should have the following:

- Tutorial project
- `Tutorial/bpel/tutorial.bpel` that you created in Part 1

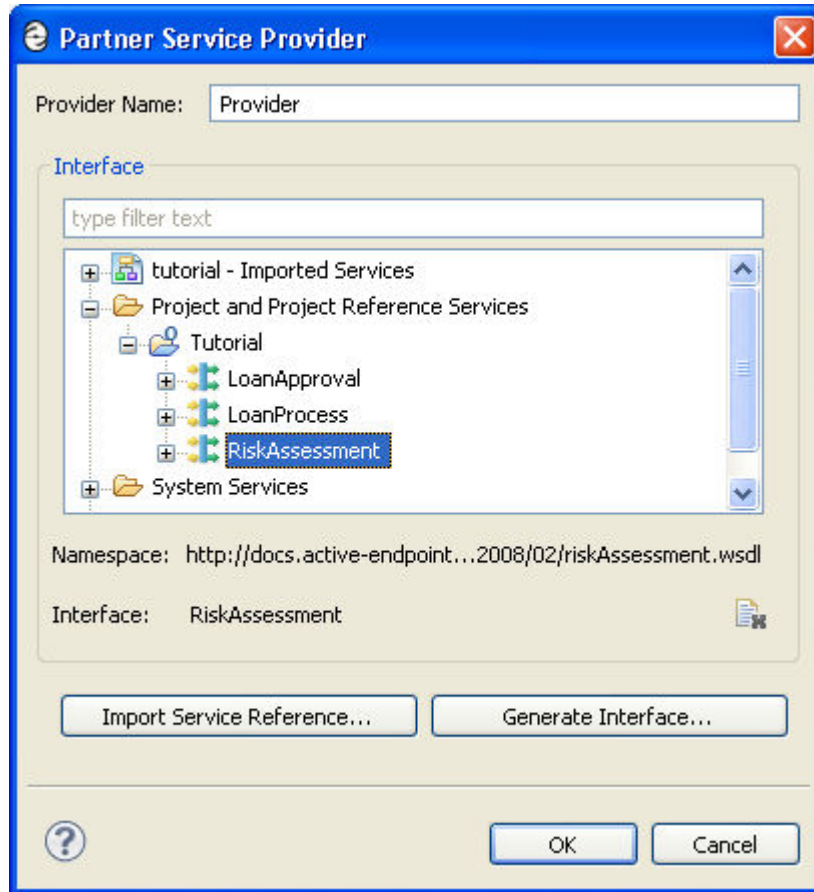
After completing Part 4 of the tutorial, you will be able to:

- Create the risk assessment and loan approval partner service participants.
- Create the risk assessment activity.

### Step 1: Create the Risk Assessment Service Provider

Recall that the process receives a request for a loan approval and is evaluated by the risk assessment service. This service checks the customer risk level and loan amount.

1. Ensure your file, `tutorial.bpel`, is open in the Process Editor.
2. In the Participants view, right-mouse click on Partner Service Providers and select New Partner Service Provider.
3. Name the service provider `RiskAssessment`.
4. Notice that the Interfaces tree displays the `LoanProcess` port type already in use for the process. Expand Project and Project Reference Services to display available port types, and select `RiskAssessment`, as shown.



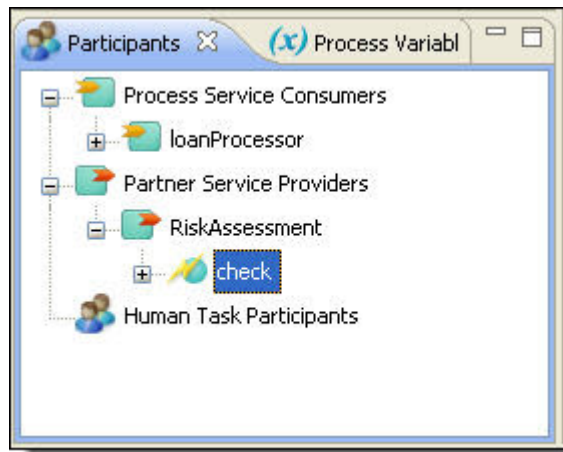
5. Click **OK**.

## Step 2: Create the Invoke Risk Assessor Activity

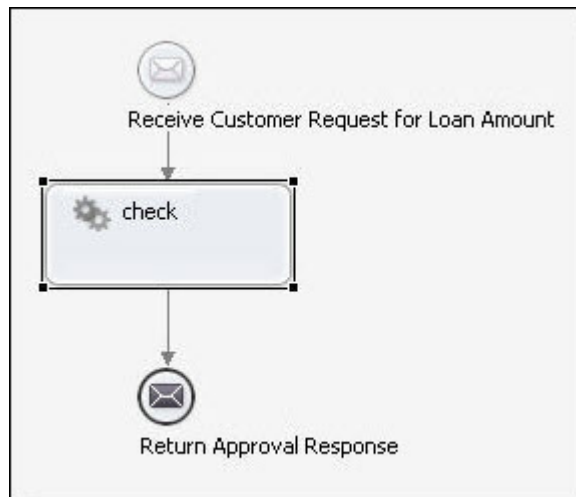
The process requires invocation of the risk assessment service. Here's a shortcut:

1. In Participants view, expand `RiskAssessment`.
2. Select the check operation, as shown in the example.

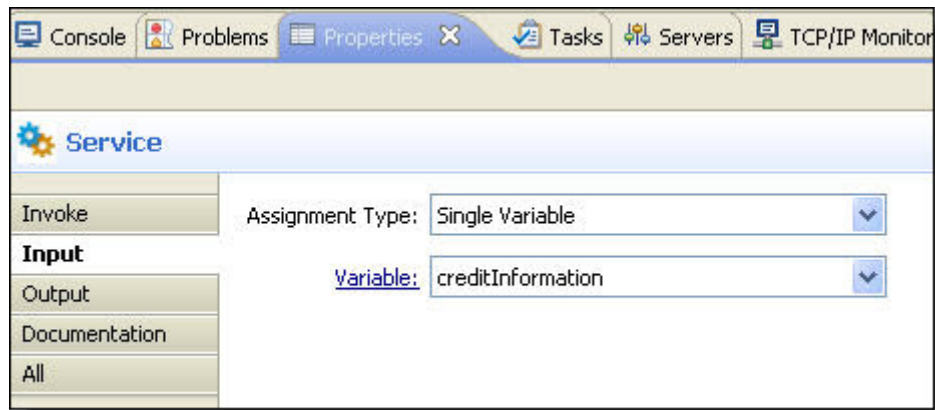




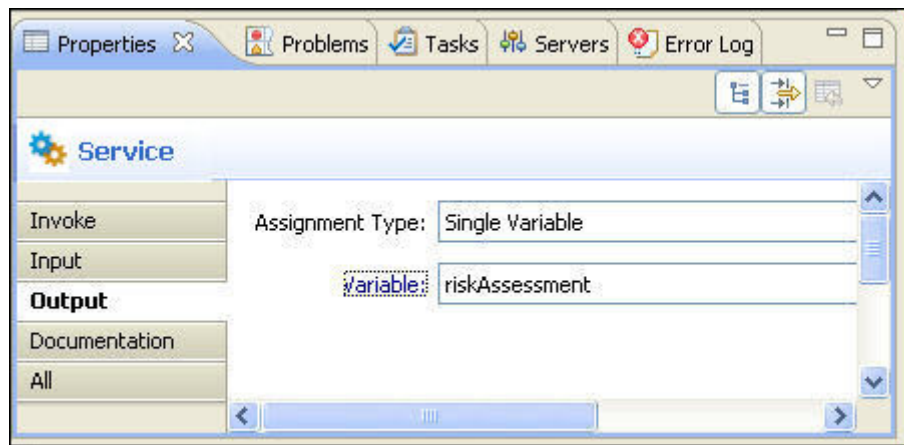
3. Drag the check operation to the Process Editor, and place it between the receive and reply, as shown.



4. A new invoke activity is automatically created, named check. In the Properties view, change the Activity Name to *Invoke Risk Assessor*. Notice that the participant and operation are automatically filled in.  
From the Color property palette, select Blue. Color adds a visual difference between participants.
5. Select the Input tab.
6. In the Assignment Type field, select Single Variable.
7. From the Variable picklist, select creditInformation, as shown.



8. Select the Output tab.
9. In the Assignment Type field, select Single Variable, if needed.
10. From the Variable picklist, select New Variable.
11. In the New Variable dialog, name the variable `riskAssessment`, as shown.



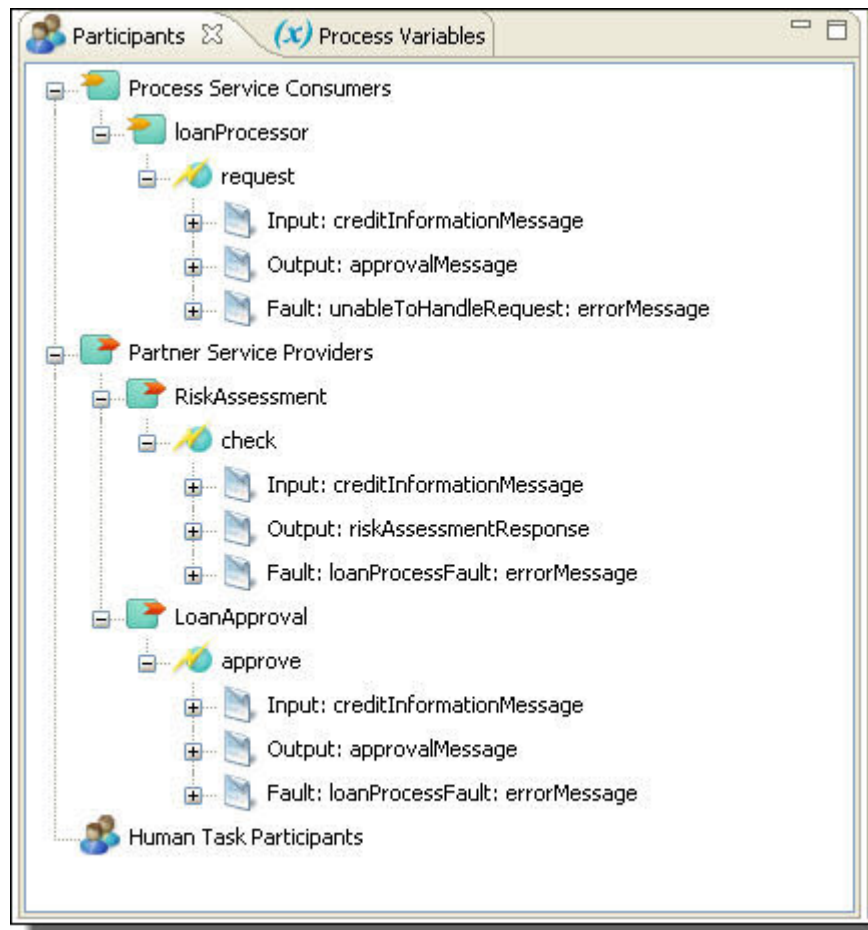
Tip: Receive and Reply activities can also be created automatically (like these invoke activities) by dragging a process service consumer operation to the Process Editor.

### Step 3: Create the Loan Approval Service Provider

Next, create a service for the process.

1. In the Participants view, right-mouse click on Partner Service Providers and select New Partner Service Provider.
2. Name the service provider `LoanApproval`.
3. Expand Project and Project References Services to display available port types, and select `LoanApproval`.
4. Click **OK**.
5. Save your file.

The Participants view should look similar to this:



You will create the invoke loan approval activity later.

## Part 5: Adding Process Activities and Properties

You have now completed parts 1 through 4 of this tutorial.

In the Project Explorer view, you should have the following:

- Tutorial project
- `tutorial.bpel` that you created in Part 4

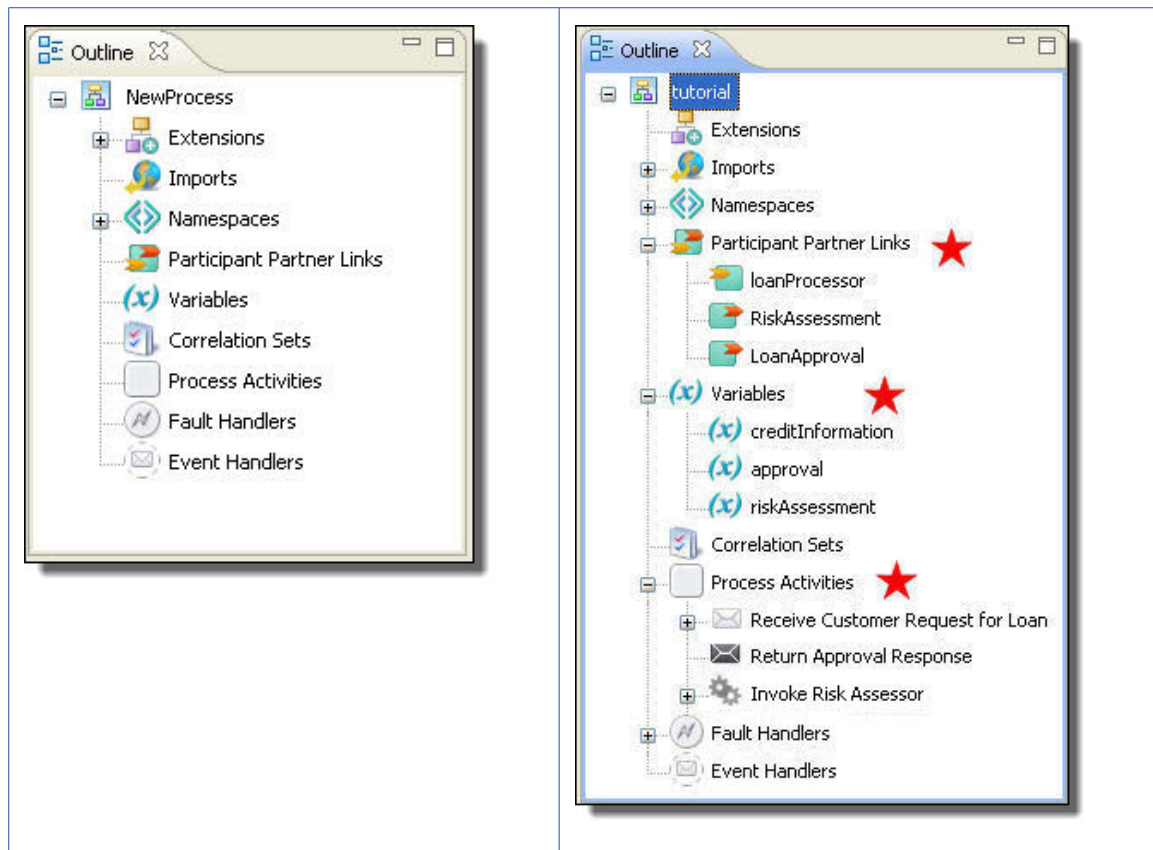
After completing Part 5 of the tutorial, you will be able to:

- Use the Outline tab to see your process in a hierarchical view.
- Rename a namespace prefix.
- Create an If activity.
- Create if conditions using the Expression Builder.
- Understand the Create Instance attribute for the Receive activity.

### Step 1: Use the Outline Tab to Add and Select Building Blocks

The Outline tab (also called a "view"), shown displays all major components of a BPEL process. Here are two Outline views. The one on the left is the default for a new process, containing only the parent nodes. Your

outline should look like the one on the right. By starting your process using participants, you have automatically added a namespace, participants, variables, and several activities.



1. In Outline view, select Correlation Sets.
2. Click and hold the Correlation Sets node, drag it to the bottom of the outline, and move it slightly until you see a black bar, indicating you can drop it. The loan approval process does not include correlation sets, so you can move this node out of the way.



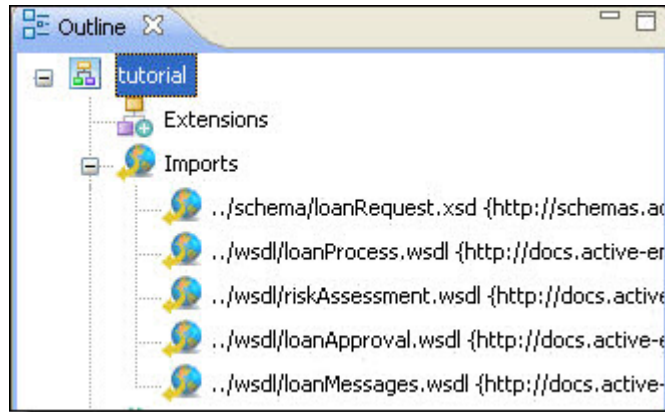
Tip: You can move items up and down to reorganize them. You can right-mouse click on a node to add new items of the same type.

### Step 2: Rename a Namespace Prefix

Each WSDL and schema file declares one or more namespaces that provide a context for messages, port types, and other definitions. The namespaces that are declared in the schema and WSDL files that your process references must also be declared in the process itself.

Because one namespace can have many WSDL files associated with it, Process Developer identifies the WSDL file by using the `Imports` function. The import is automatically added when you create a participant, and a default namespace prefix is also added, which you can rename to make more meaningful.

1. In Outline view, expand Imports to see the location of the WSDL.



2. Expand Namespaces and notice the new namespaces. The `loan` prefix and the `loanMessages` prefix are two of the namespaces added. Process Developer creates namespace prefixes based on WSDL filenames and adds them to the process. The `loanProcess.wsdl` generates the `loan` prefix the `loanRequest.xsd` file and `loanMessages` for `loanMessages.wsdl` file.
3. Notice the prefixes for all imported WSDLs, which will appear in variable expression that you will create in the process:

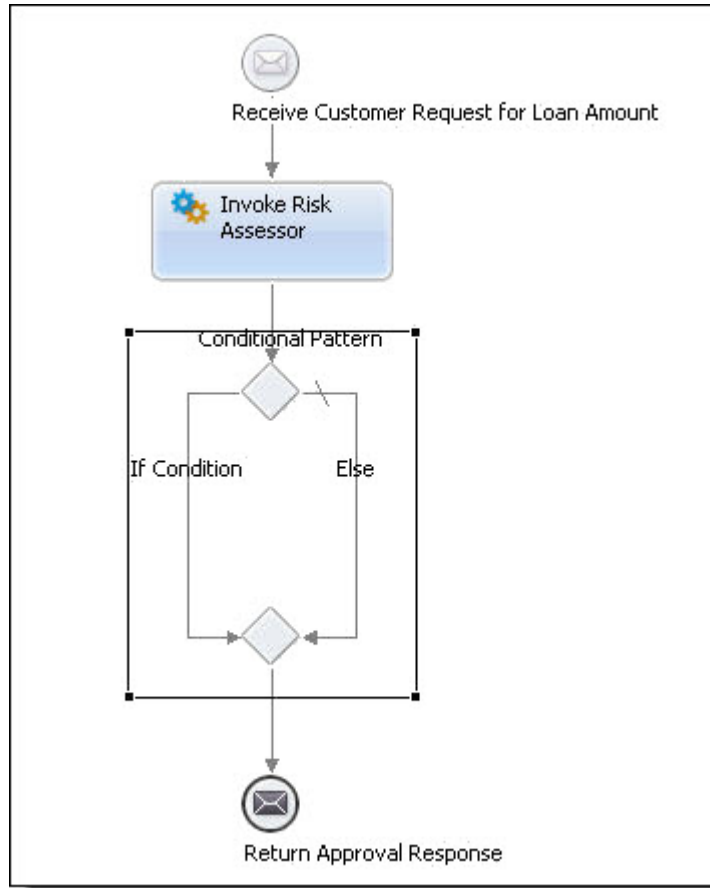
WSDL/XSD file from the Namespace URI	Prefix
loanRequest.xsd	loan
loanMessages.wsdl	loanMessages
loanProcess.wsdl	loanProcess
riskAssessment.wsdl	riskAssessment
loanApproval.wsdl	loanApproval

Tip: All new processes contain common namespaces. The `xsd` namespace defines the location of the XML schema. The `bpel` namespace defines the location of the BPEL schema. Process Developer adds other namespaces used internally by the process.

### Step 3: Adding Programming Logic with an If Activity

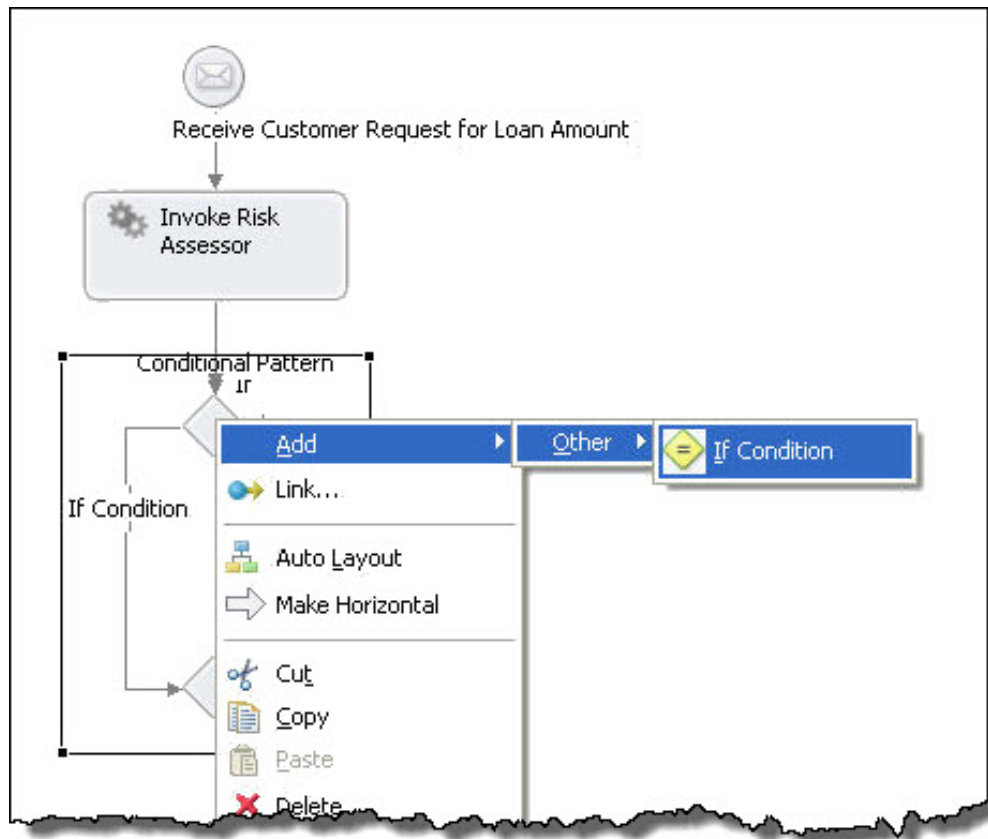
The risk assessment determines who is a high risk or low risk, or who needs more review. These conditions will be represented using the If control flow activity.

1. From the Control Flow drawer of the palette, select a Conditional Pattern activity and drop it just below the invoke activity as shown.

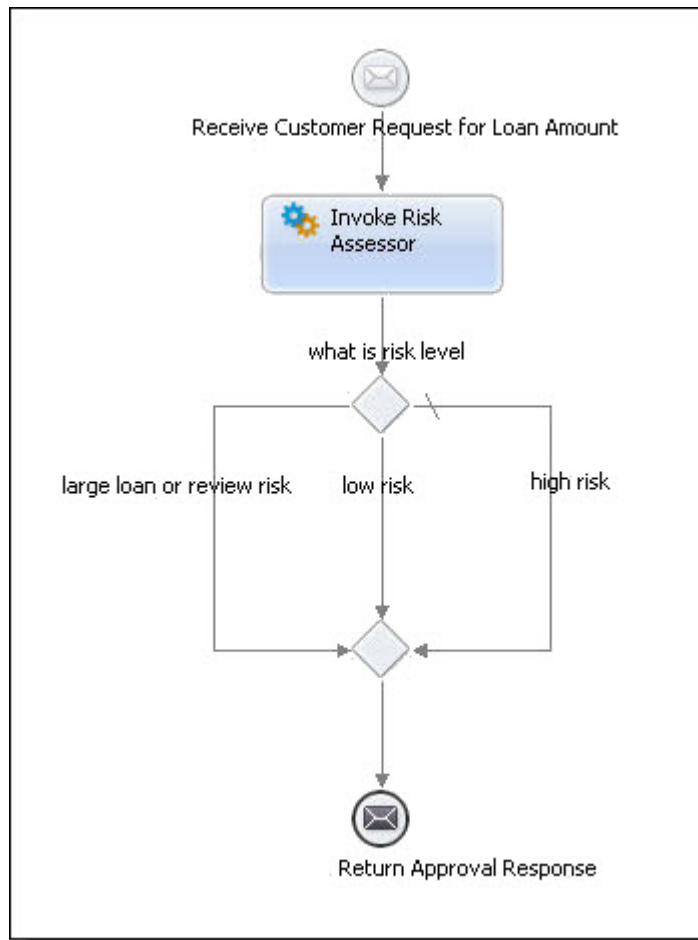


Note that the activity is already structured with If and Else conditional paths. The else condition is indicated with a slash at the beginning of its path.

2. We need to add an Else If condition. Select the starting diamond of the If, and right mouse-click to select **Add > Other > If Condition**, as shown.



3. Select the first *If Condition* label and rename it `large loan or review risk`.  
Tip: Click once to select the label and then click the text to edit it.
4. Select the next *If Condition* label and rename it `low risk`.  
Be sure to label the two if conditions in order, left to right. The left-most condition is evaluated first at runtime, as you will see later.
5. Select the *Else* label and rename it `high risk`.
6. Select the *Conditional Pattern* label and change it to `what is risk level`.  
Here is what your process should look like:



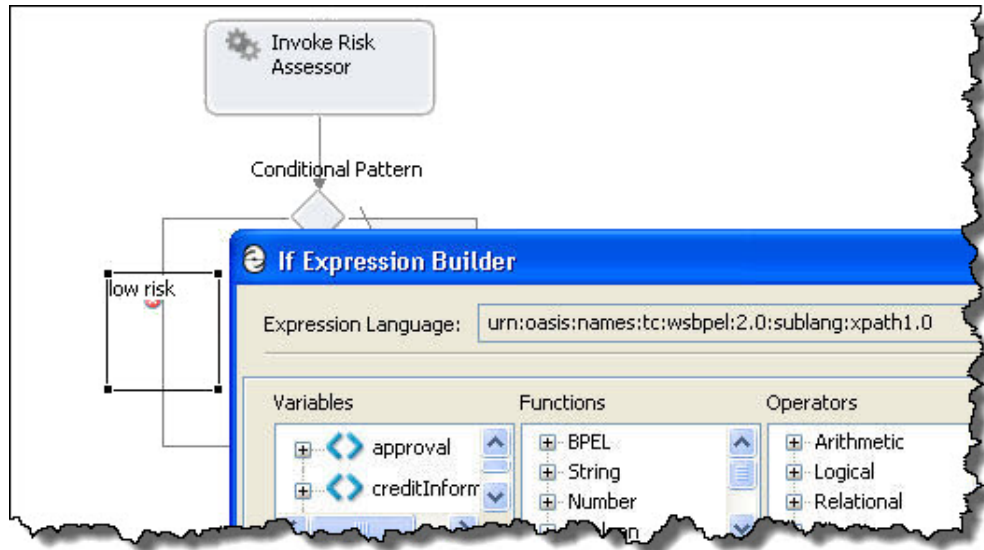
7. Save your file. More errors are reported, which will now be fixed.

#### Step 4: Create Conditional Expressions

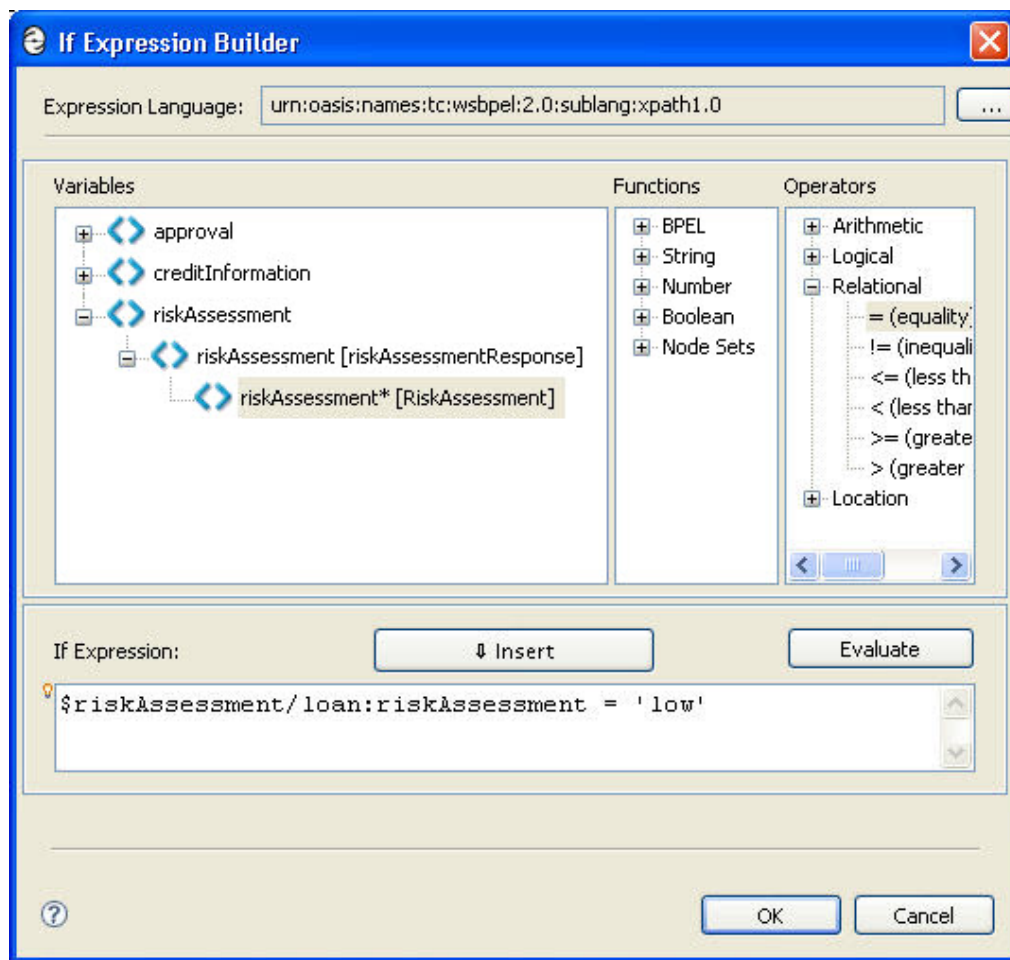


Begin by adding conditional processing for each if condition path.

1. Double-click the `low risk` If activity (not the label) to open the **If Expression Builder**, as shown.

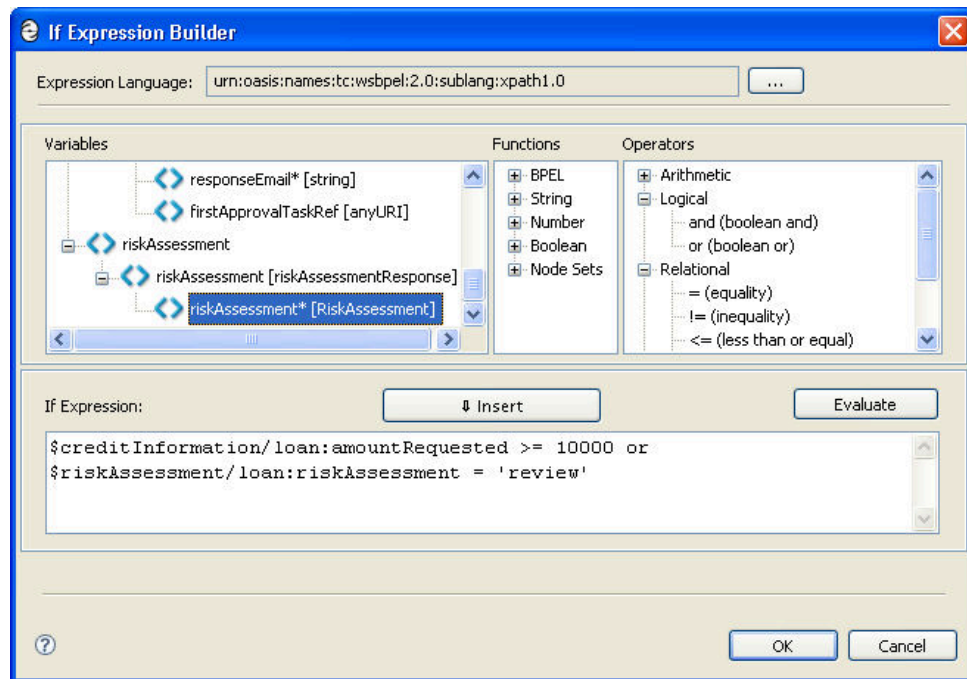


2. In the Builder dialog, expand the `riskAssessment` variable and double-click the `riskAssessment` part to add an expression to the If Expression box.
3. Complete the expression as follows: `$riskAssessment/loan:riskAssessment = 'low'` This is shown in the following example.



4. For the large loan or review risk condition, create an expression for two conditions:
  - a. Expand the `creditInformation` message, and double-click the `amountRequested` part.
  - b. Complete the expression by adding the large loan amount and the risk assessment review condition, as shown.

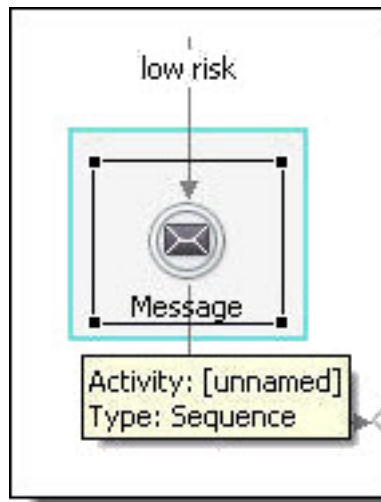
```
$creditInformation/loan:amountRequested >= 10000 or
$riskAssessment/loan:riskAssessment = 'review'
```



#### Step 5: Create the Reply for the Low Risk Condition

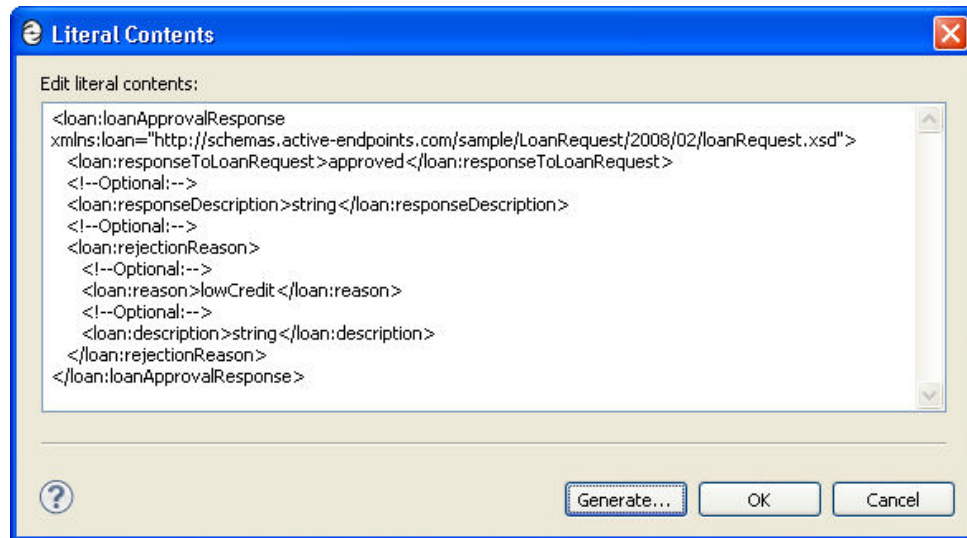
If the customer requests less than \$10,000 and is a low risk, the customer is approved for a loan. Here's how to create a reply with this message.

1. From the Throw Event drawer of the palette, drag a Message to the blue bounding box of the low risk condition as shown.



2. Fill in the reply properties in the Properties view as follows:
  - a. Activity Name: Approve
  - b. Participant: loanProcessor
  - c. Operation: request
3. Select the Data tab of the Properties view.

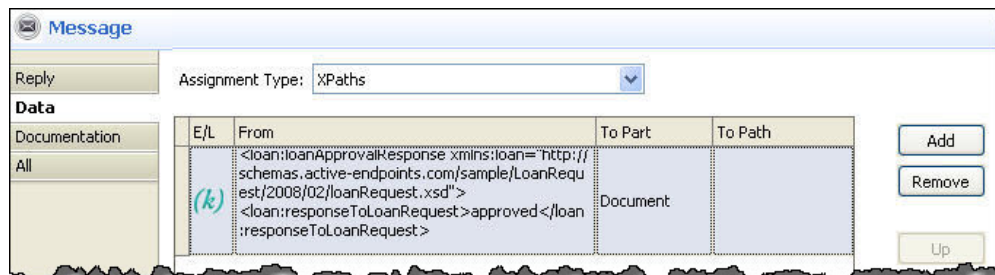
4. Note that the Assignment Type is *XPaths*. XPath expressions allow you to select nodes from a variable for assignment. You will now map a literal expression to the reply variable, as follows.
  - a. The table structure on this data tab allows for several XPath expressions. We need only one expression. Select **Add**. (The button is to the right of the table.)
  - b. In the *E/L* column, select the *f(x)* (Expression) entry and change it to *(k)* Literal expression.
  - c. Click in the *From* column and select the **Dialog (...)** button to open the **Literal Contents** dialog.
  - d. Select **Generate**, and in the XML Data Wizard, notice that the matching data type, `loanApprovalResponse`, is selected as the Root Element. The element-based approval message is a single part element.
  - e. Select **Finish**. The Literal Contents looks like the following:



- f. Replace the `loan:responseDescription` string with the following: Your excellent credit has earned you automatic approval for your loan.
- g. Delete the entire `<loan:rejectionReason> </loan:rejectionReason>` element. The literal expression will be:

```
<loan:loanApprovalResponse xmlns:loan=
  "http://schemas.active-endpoints.com/sample/LoanRequest/
    2008/02/loanRequest.xsd">
  <loan:responseToLoanRequest>approved
</loan:responseToLoanRequest>
  <loan:responseDescription>Your excellent credit has earned you automatic
  approval for your loan.
</loan:responseDescription>
</loan:loanApprovalResponse>
```

The Data tab should look like the following.



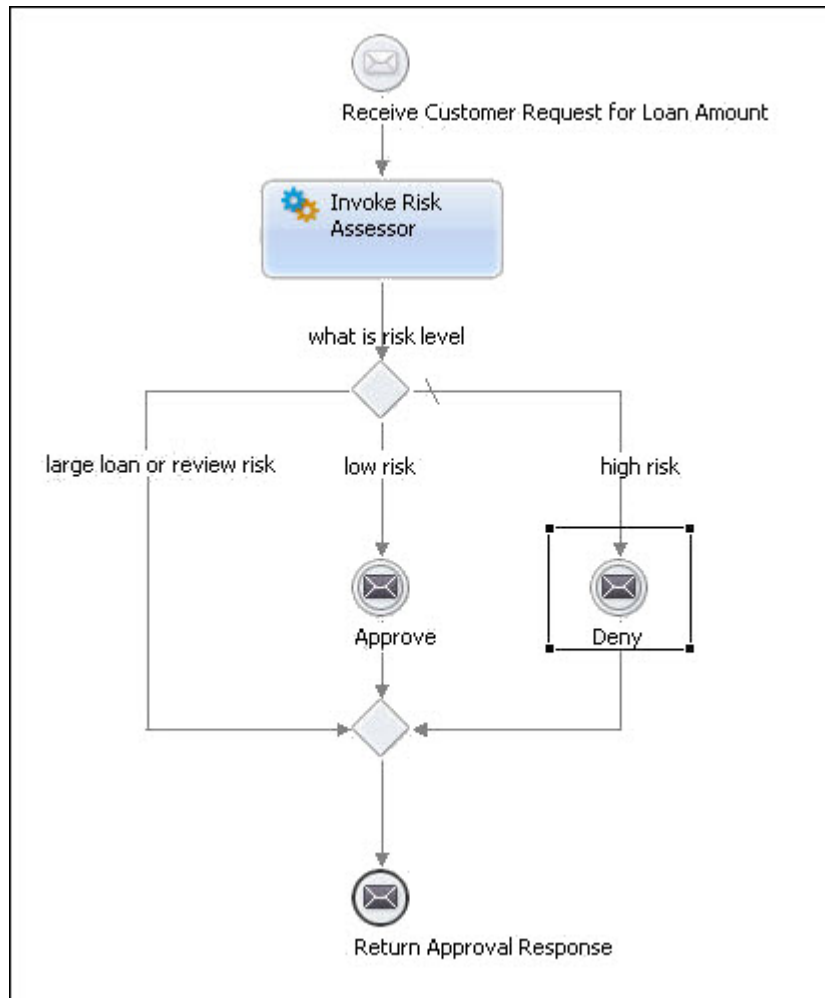
### Step 6: Create the Reply for the High Risk Condition

The reply for the high risk customer is created with almost identical steps used for the low risk customer. You can use a shortcut and make a copy of the approve reply to create the deny reply.

1. Copy the Approve reply and paste it into a blank area of the canvas. The reply is pasted into the top-left corner.
2. In the Properties view, rename the Activity Name to *Deny*. The Participant and Operation are already filled in.
3. In the Data tab, select in the *From* cell to display a dialog (...) button, and then select it.
4. Select Generate, then Finish.
5. Edit the literal contents for the denied loan. The message should read as follows:

```
<loan:loanApprovalResponse xmlns:loan=
  "http://schemas.active-endpoints.com/sample/LoanRequest/
    2008/02/loanRequest.xsd">
  <loan:responseToLoanRequest>
    declined
  </loan:responseToLoanRequest>
  <loan:responseDescription>We are sorry, this application
    falls outside of our credit risk guidelines.
  </loan:responseDescription>
  <loan:rejectionReason>
    <loan:reason>lowCredit</loan:reason>
    <loan:description>low credit score
  </loan:description>
  </loan:rejectionReason>
</loan:loanApprovalResponse>
```

6. Drag the Deny reply to the bounding box in the high risk condition. Your process should now look like the following.

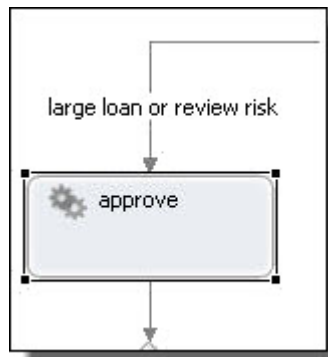


### Step 7: Create the Invoke Loan Approver Activity

You have one more path to complete for the If activity: the case where the loan request is large or the customer's risk level could not be determined by the risk assessment service.

Recall that you have already created two partner services for the process and are already using the risk assessment service via the risk assessment invoke activity. Now you will use the loan approval service.

1. In the Participants view expand Partner Service Providers to view the LoanApproval partner.
2. Expand LoanApproval to view the approve operation.
3. Drag the approve operation to the bounding box of the large loan or review risk condition. A new invoke activity is created, as shown.



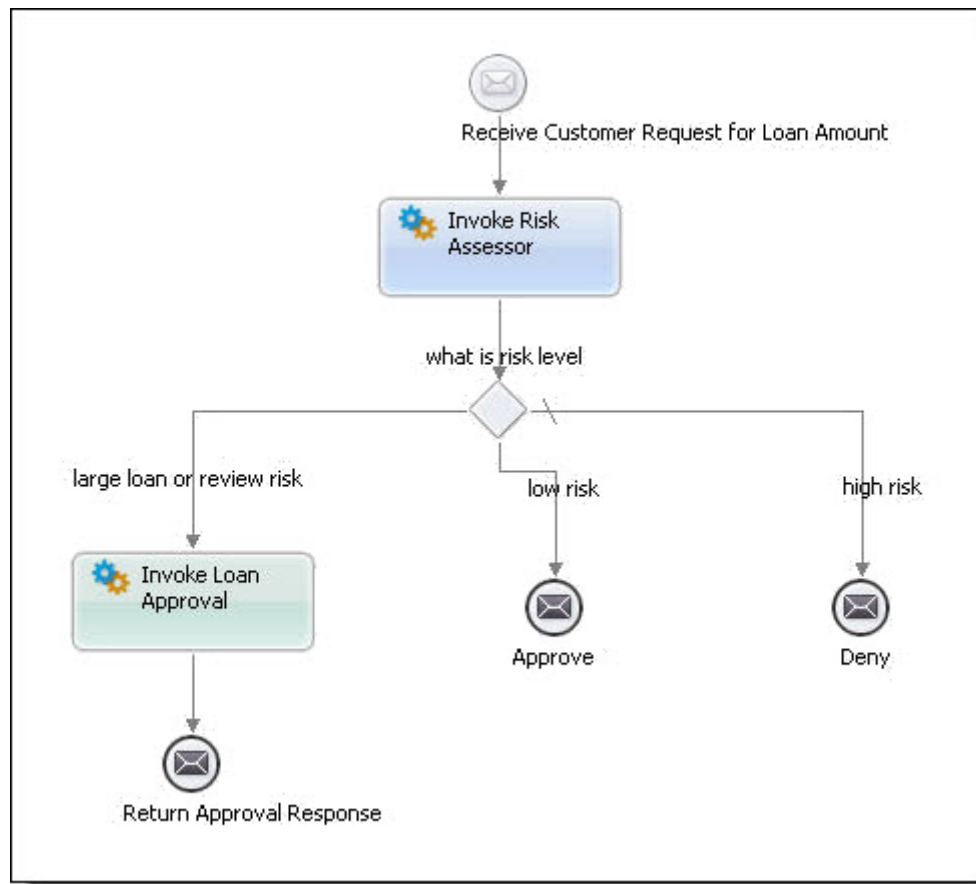
4. In the Properties view, name the activity `Invoke Loan Approval`.
5. Select Green for the activity color.
6. On the Input tab, in Assignment Type, select Single Variable.
7. For the Variable, select `creditInformation`.
8. On the Output tab, in Assignment Type, select Single Variable.
9. For the Variable, select `approval`.

#### **Step 8: Add the Loan Approval Reply**

Select the Return Approval Response reply and drag it into the bounding box containing the new invoke activity. Be sure to move the activity very close to the bottom of the invoke activity to place it in sequence inside the bounding box.

Notice that the process now has three discrete end activities, clarifying the process flow.

Your completed process definition should look like the following.



### Step 9: View the Create Instance property for the Receive activity

The Receive is the activity that kicks off the process, so you want to ensure that the Create Instance property is enabled. Process Developer automatically adds the Create Instance property to the Receive (or other start) activity that you add to the process.

1. In the Process Editor, select the Receive activity.
2. In the Properties view, notice that a checkmark was added to the Create Instance box.

### Step 10: Save Your File and Validate Your Process

1. Save your file. When you save it, it is automatically validated for BPEL.
2. View Problems view, stacked next to Properties view. There should be no errors listed.  
If any errors are listed, double-click the error to go to the source of the error.

**Tip:** You can compare your file to `tutorialCompleted.bpel` to discover any differences. To open a completed version of the tutorial, select **File > New > Orchestration Project**, name the project, click **Next**, and then select the Template *Tutorial Completed*.

**Note** that the `tutorialCompleted.bpel` file has one difference: The sequence surrounding the process was removed by using the Ungroup right-mouse menu option.

You have completed the main process definition.

## Part 6: Adding Fault Handling

To start at the beginning of the tutorial, see [“Part 1: Starting a New Process” on page 35](#).



Fault handling in a BPEL process is reverse work, undoing the partial and unsuccessful work of a scope in which a fault has occurred. When a fault occurs, normal processing is terminated, and control is transferred to the associated fault handler.

So far in the tutorial, you have completed a BPEL process definition that contains all the steps for normal processing. Now you will add a fault handler to handle a service invocation fault.

If the loan process throws a fault, it terminates the process using a standard fault, and turns over control to the fault handler activity.

In the `loanprocess.wsdl` file, there is a fault name and a fault message defined for the WSDL's operation, namely the request operation. The loan process uses the fault name and message in defining fault handling activities for the assessor and approver services.

In the Project Explorer view, you should have the following files:

- `tutorial.bpel` that you created in Part 2
- `tutorialCompleted.bpel`, which is a completed version of the file (optional)

After completing Part 6 of the tutorial, you will be able to:

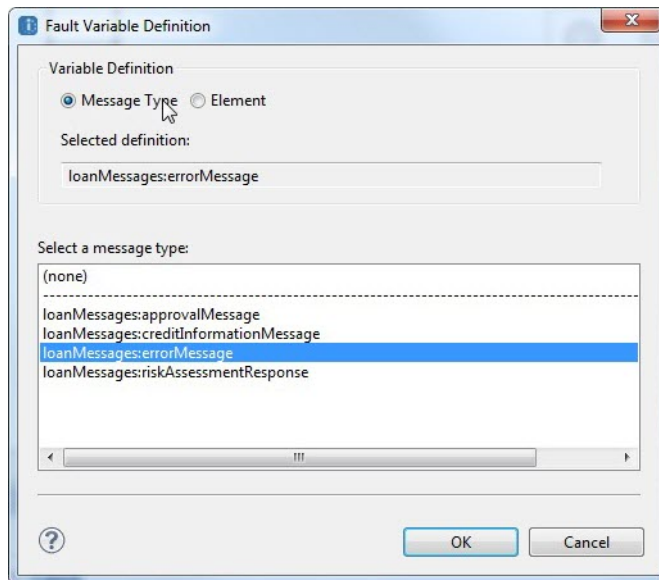
- Add a Catch fault handler for the process.
- Add a fault variable.
- Add a fault handling activity to the Catch handler.

#### **Step 1: Add a Catch Fault Handler and Fault Variable**

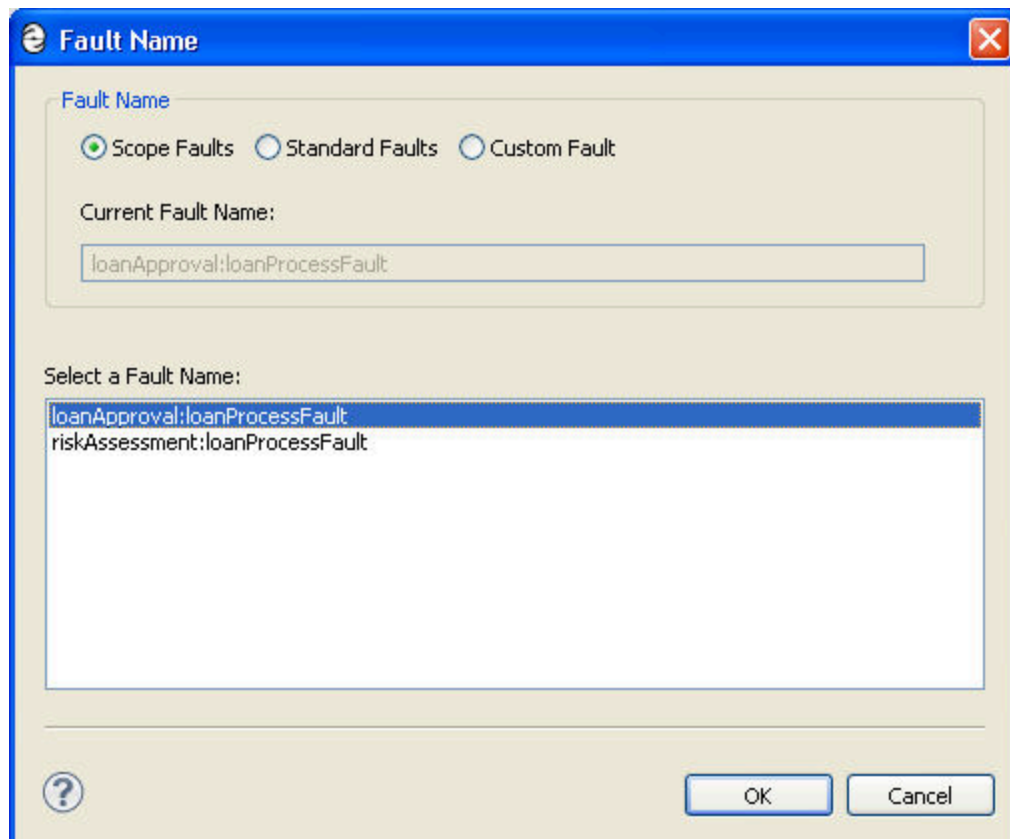
When a fault handler receives an inbound fault message, it assigns the fault message to a variable before proceeding to perform an activity enclosed by the catch.

1. Click on the Fault Handlers tab of the Process Editor.
2. From the Catch Event palette, drag an Error activity to the canvas.
3. Select the activity.
4. In the Properties view, select Fault Variable Definition, and click the dialog (...) button at the end of the row.
5. Fill in the **Fault Variable Definition** dialog as follows and click OK:
  - a. Select the Element radio button.

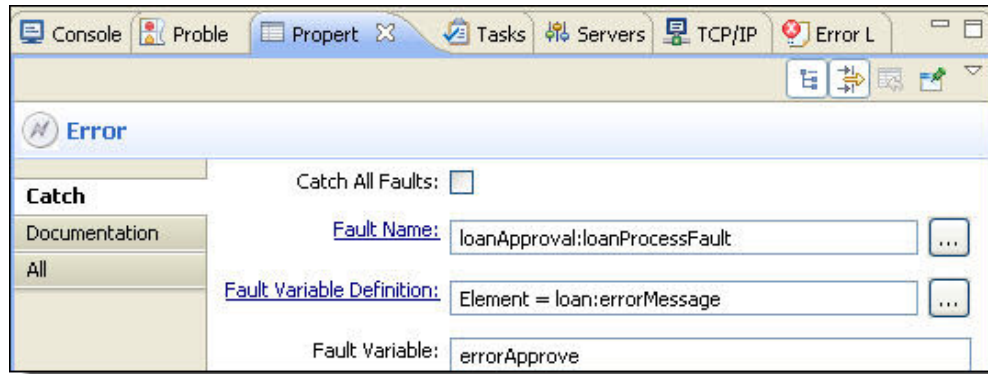
- b. From the list of messages, select `loan:errorMessage`.



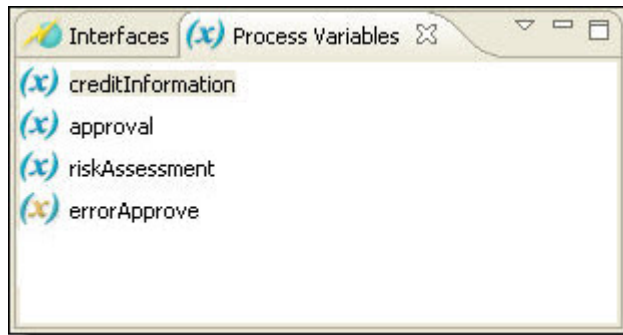
6. In the Properties view, select Fault Name, and click the dialog (...) button at the end of the row.
7. In the Fault Name list, notice that the WSDL fault name associated with both the assessor and approver services is the same, but each is in a different namespace. For convenience, the WSDL for each service used the same fault name, but usually if you are orchestrating two different services the fault names would be different. Select the `loanApproval:loanProcessFault`, as shown in the example.



8. In the Fault Variable field, type `errorApprove`, which is the name we will associate with the fault variable definition. The properties for the Catch activity should look like the following example:



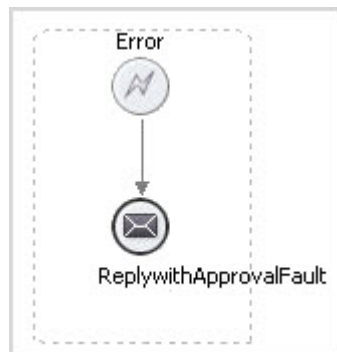
Notice that a new variable named `errorApprove` was to Process Variables view. This variable is exclusively for fault handling, and this is shown using an icon that differs from normal process variables.



## Step 2: Add a Fault Handling Activity

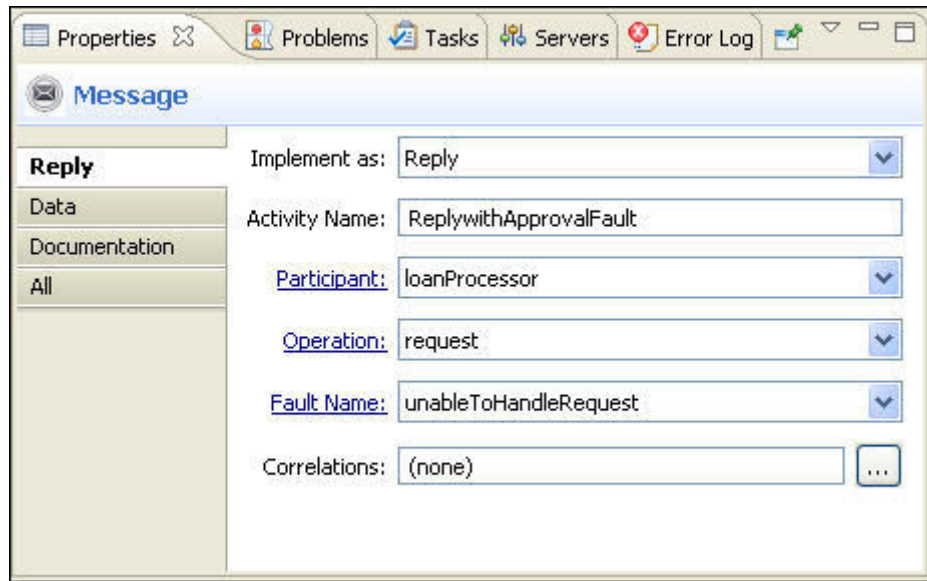
When a fault is caught, the fault handler must execute an activity. You will add a Reply activity to tell the customer that the process was unable to handle the request.

1. From the Throw Event palette, drag a Message activity into the Catch handler and name the activity `ReplywithApprovalFault`, as shown.

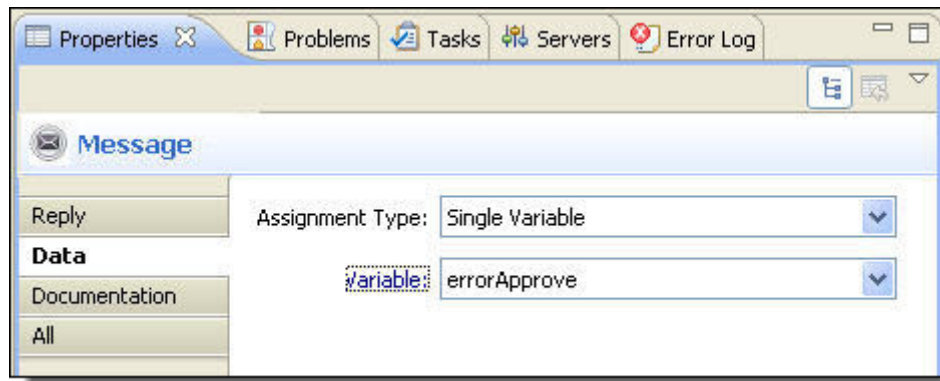


2. Fill in the properties for the reply activity to handle the fault, as shown in the example. You must select them in the following order:
  - a. Participant (`loanProcessor`)
  - b. Operation (`request`)

c. Fault Name (unableToHandleRequest)

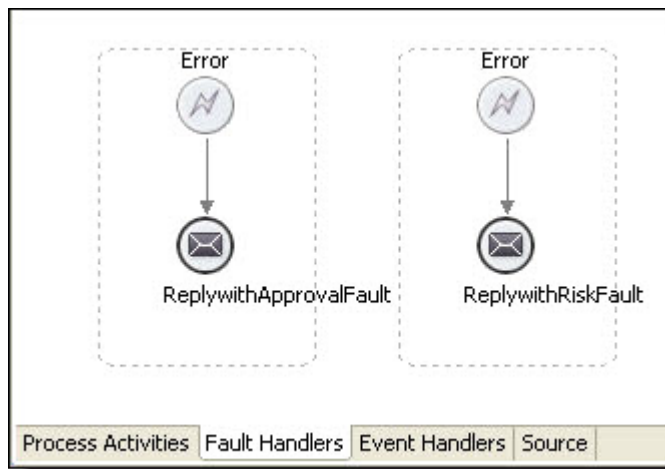


3. On the Data tab, select the new variable (errorApprove).



4. Add a similar Fault handler for the risk assessment service by selecting the same fault variable definition and the new fault name `riskAssessment:loanProcessFault` and naming the fault variable `errorRisk`. Name the Reply `ReplywithRiskFault`.

Your Fault Handlers view should be similar to the following:



## Part 7: Adding Compensation and Correlation

To start at the beginning of the tutorial, see [Chapter 3, "ActiveVOS Tutorial" on page 34](#).

Compensation is the process of reversing or providing an alternative for a successfully completed activity, especially when a fault occurs. Compensation restores data to what it was before the activity work was done.

Correlation is a construct for keeping track of a group of messages that belong together in one particular business partner interaction. Correlation matches messages and interactions with the business process instances they are intended for.

The loan approval tutorial does not require correlation or compensation.

For more information, see "Compensation and Correlation" elsewhere in this help.

## Part 8: Simulating the Process

To start at the beginning of the tutorial, see [Chapter 3, "ActiveVOS Tutorial" on page 34](#).

If you have followed all tutorial parts so far, you have completed a BPEL process definition that contains all the steps for normal and fault processing.

Now you will turn from the design phase to the testing phase by simulating execution of your process.

In the Project Explorer view, you should have the following files:

- `tutorial.bpel` that you created in Part 2
- `tutorialCompleted.bpel`, a completed version of the file (optional)

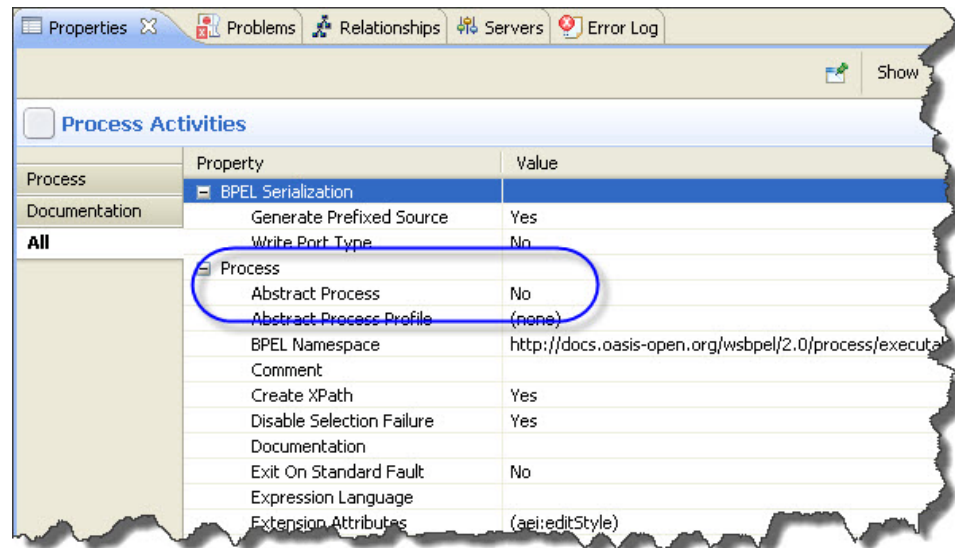
After completing Part 8 of the tutorial, you will be able to:

- Complete the prerequisite checklist for simulation.
- Add sample data files for WSDL message parts.
- Display data values for process variables.
- Simulate execution of a BPEL process.
- Test all execution paths in a process by overriding default data values.

### Step 1: Complete the Prerequisite Checklist for Simulation

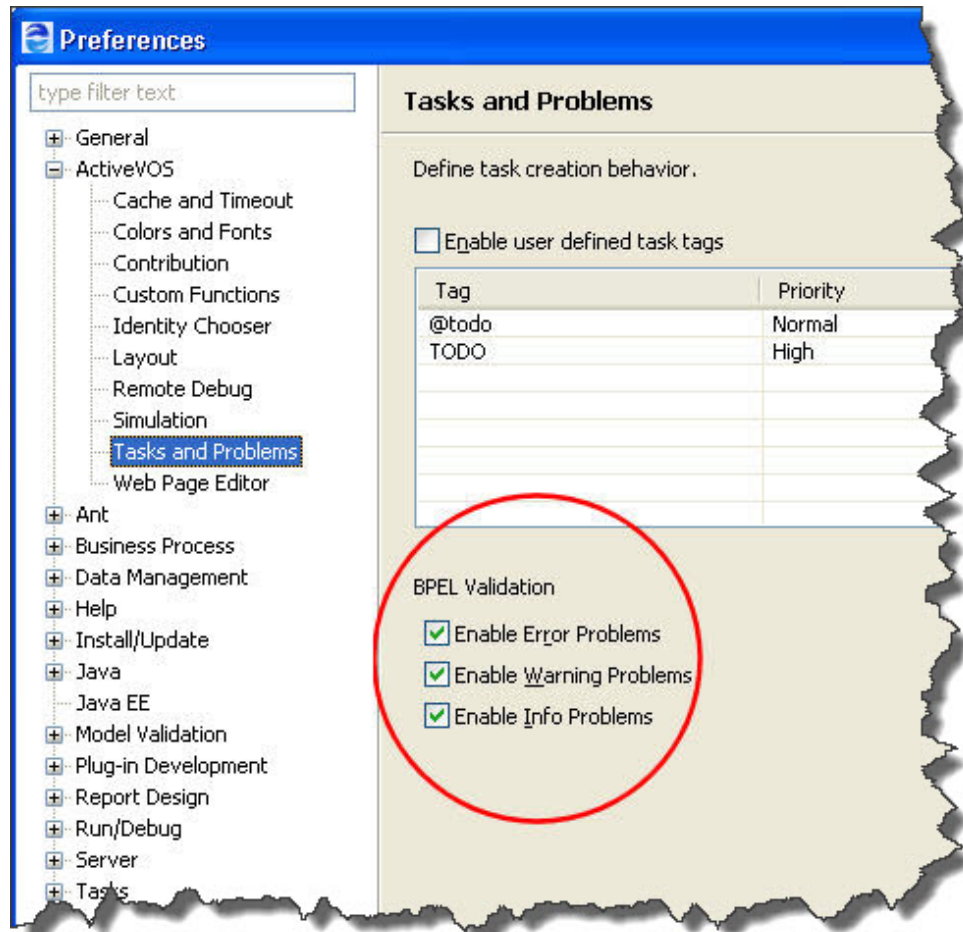
Process Developer validates your BPEL process before you execute it, adding validation tasks to the Problems view for you to complete. The tasks are broken down into errors, warnings, and information. You must ensure your process is executable, and you must complete all error tasks before running your process.

1. Open `tutorial.bpel`.
2. Ensure that the Abstract Process property is set as follows:
  - a. Select Properties view.
  - b. Click on an empty part of the Process Editor (or select *tutorial* from the Outline view).
  - c. Select the All tab in the Properties view and note that Abstract Process is set to No to indicate that the process should be validated for execution.

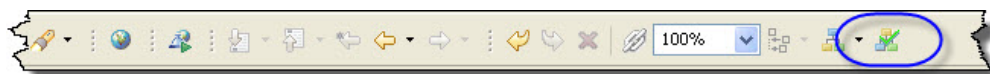


3. Ensure that Process Developer is reporting BPEL validation tasks, as follows:
  - a. Select Window > Preferences > Process Developer > Tasks and Problems.

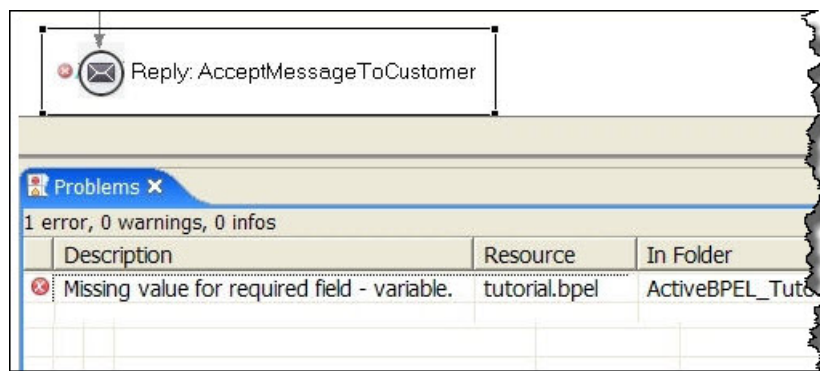
- b. All BPEL Validation settings should be selected, as shown.



4. Validate your file by selecting the Validate Process toolbar button. This action ensures that all validation tasks appear in the Problems view.



5. Select Problems view.
6. If an error task exists, such as the one shown in the following example, correct the error, and save your file.

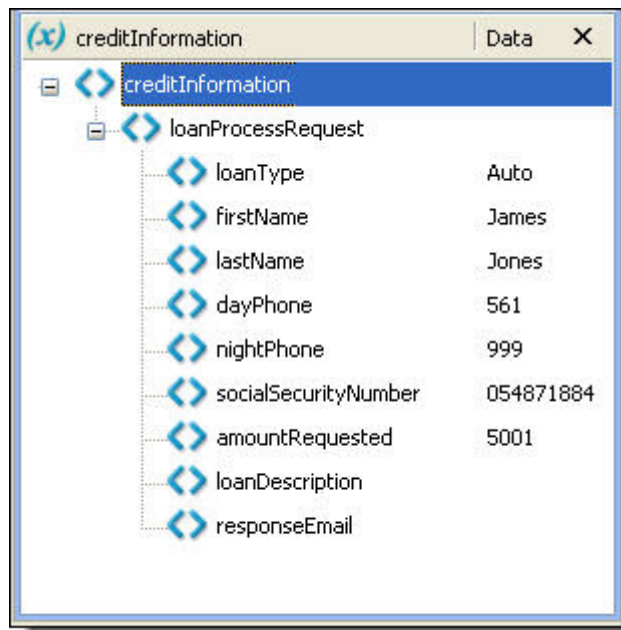


## Step 2: Load Sample Data Files for the Messages

To simulate process execution, you need to initialize process variables. Process Developer has a convenient way to provide default data values and ways to override the defaults for different test scenarios.

When you create an orchestration project, you can add sample data files into the Process Variables view (or into the Project Explorer). The sample data is registered and is available for all processes that use the messages.

1. Expand the sample-data folder in the Tutorial project. Note that several files were created for you to test with. You will load some of these files into variables for different test scenarios.
2. Select Process Variables view.
3. Open the `creditInformation` variable by double-clicking it. You can see the data type definition.
4. Right-mouse click `creditInformation` and select View Data.
5. Right-mouse click and select Load Data > From Project File.
6. Select `CredInfo_Jones5001.xml` from the sample-data folder.  
Your `creditInformation` variable should look similar to the following example.



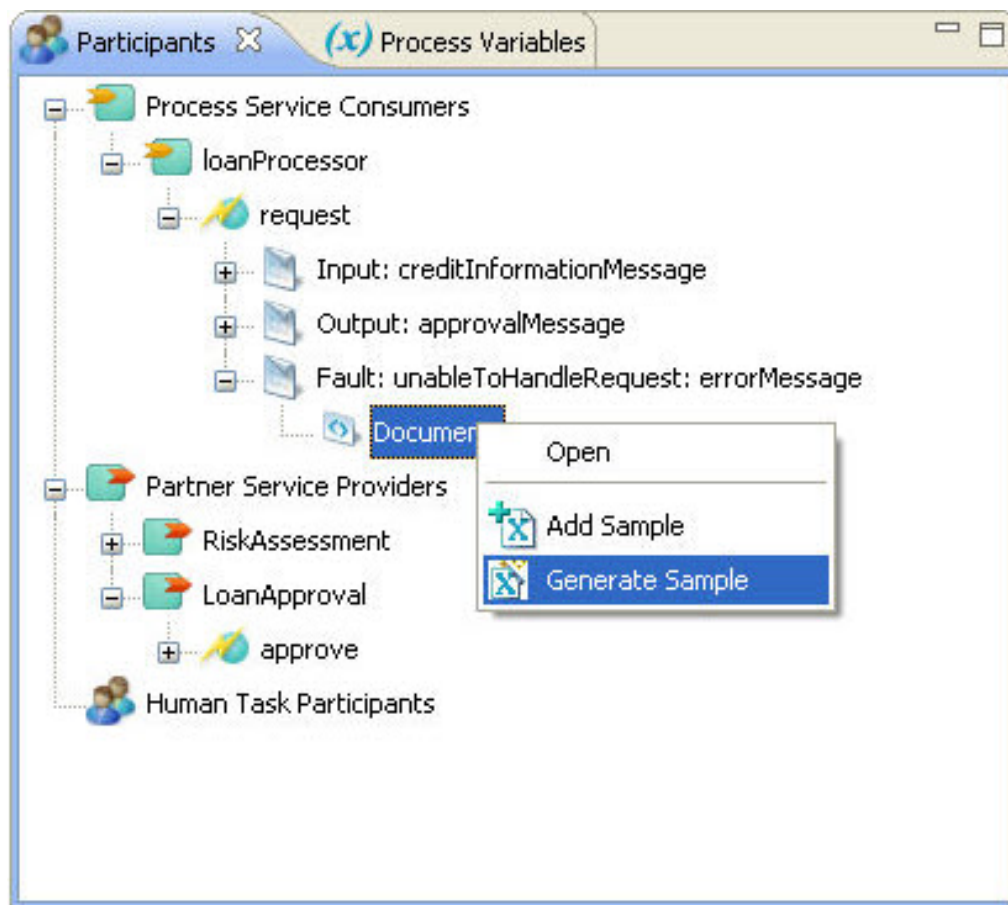
7. Load `assessment_low.xml` sample data file for the `riskAssessment` variable.

### Step 3: Generating a New Sample Data File for the errorRisk Variable

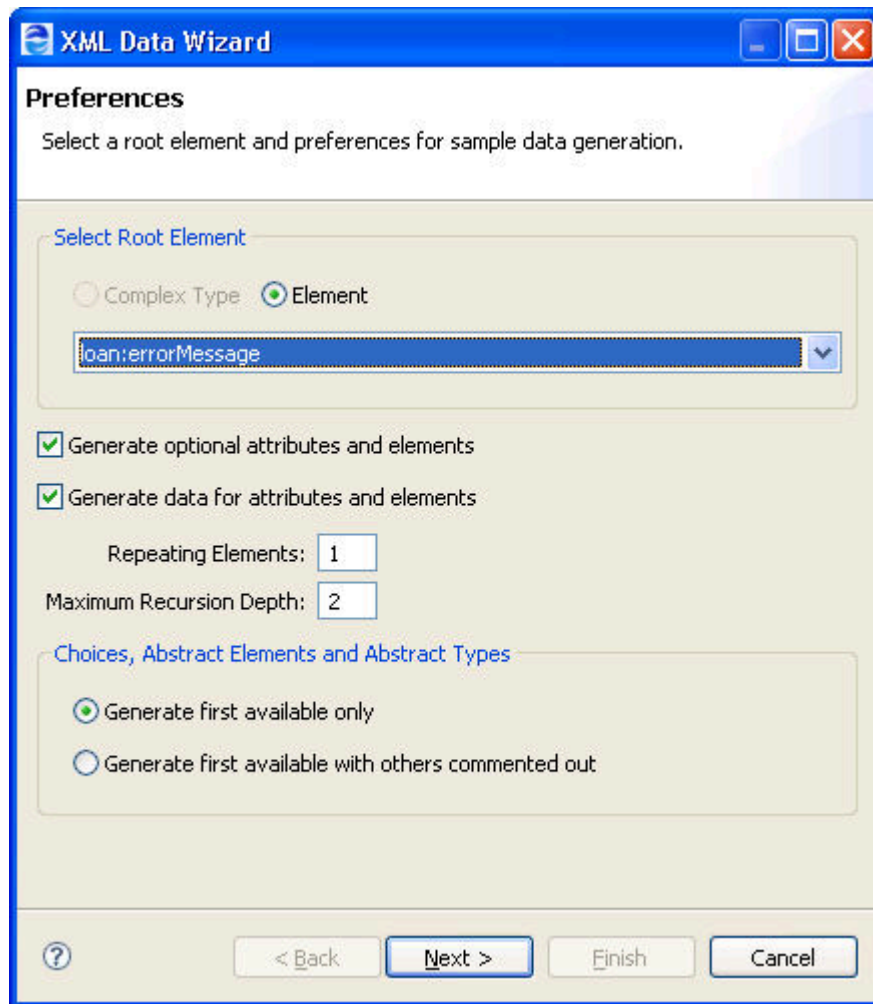
The `errorRisk` variable, as all the other variables, is defined with a schema complex type and requires namespace-qualified sample data. This means the sample data file must contain references to the data type definition. You will automatically generate a valid data file using the XML Data Wizard, as you can do with any variable to create your own samples. This step is similar to generating literal contents for the Copy Operation, as you did in Part 5 of the tutorial. This wizard uses the type definition located in `loanmessages.xsd`.

1. In Participants view, select the `loanProcessor` participant to view the request operation's messages.
2. Expand the Fault `errorMessage`.
3. Select the document part of the fault message, and from the right-mouse menu, select Generate Sample as the illustration shows.

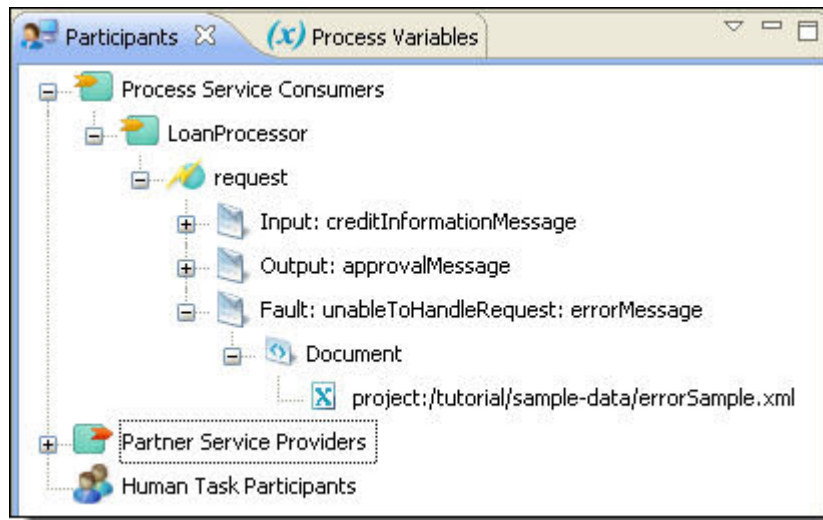




4. The definition of the message points to a schema element, which was selected by the wizard as shown in the following example:



5. Leave the remaining preferences as is, since they do not apply to the schema definition of this message, and click **Next**.
6. In the Save Results wizard page, select the `Tutorial\sample-data` folder, and name the generated sample data file `errorSample`.
7. Click **Finish**, and then expand the Document message part to see the `errorSample.xml` file.



If you wish to view the contents of the data file, right-mouse click it, and select Open. The XML file opens in the editor. Close the file when you are done.

8. In Process Variables, notice that `errorSample.xml` was loaded into the `errorRisk` variable.
9. Your open process variables should look similar to the following:

(x) creditInformation	Data
creditInformation	
loanProcessRequest	
loanType	Auto
firstName	James
lastName	Jones
(x) riskAssessment	Data
riskAssessment	
riskAssessmentResponse	
riskAssessment	low
(x) errorRisk	Data
<b>errorRisk</b>	
errorMessage	
errorCode	1
reason	string

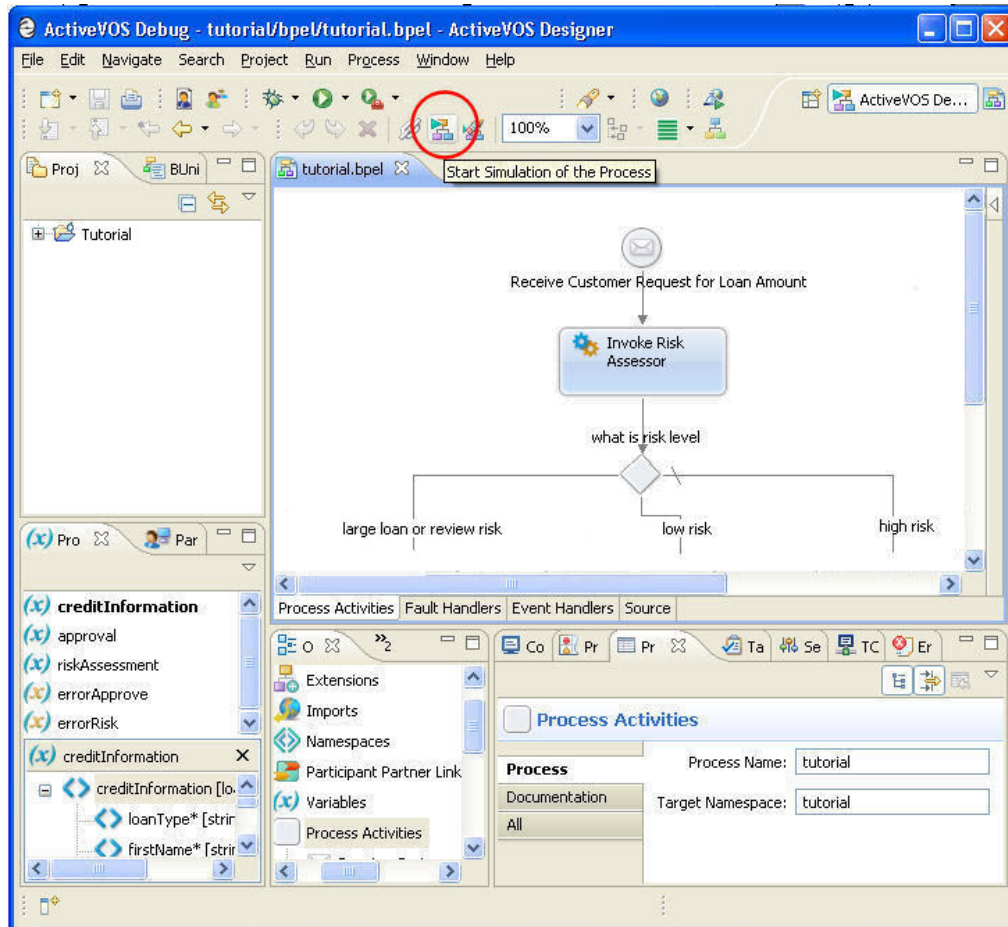
Notice the following visual information cues in the Process Variables view:

- In the Process Variables list, **bold** indicates an open variable.
- The `errorRisk` and `errorApprove` variables are two-toned, indicating they are available only as fault variables.

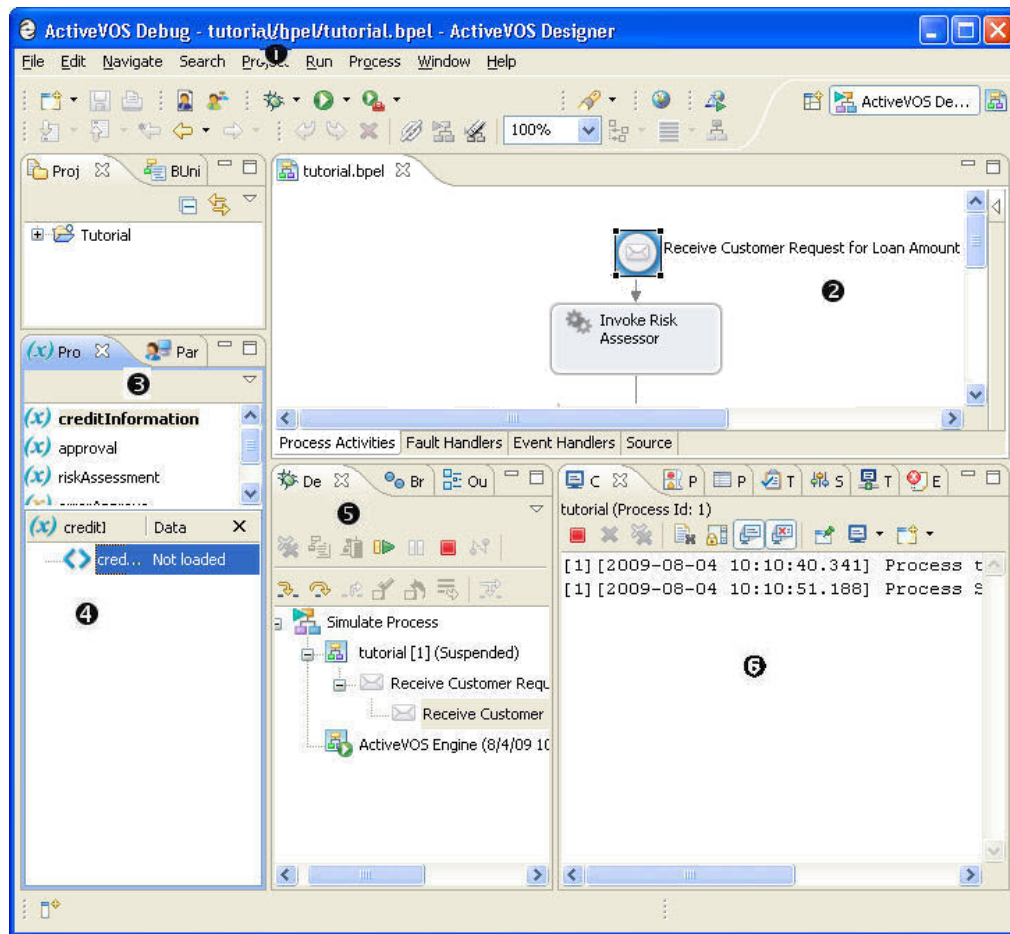
#### Step 4: Simulate Process Execution

If you have completed all previous steps in this part of the tutorial, you are ready to begin simulating execution of your process.

1. Click on a blank part of the Process Editor canvas to activate the main toolbar.
2. Select the Start Simulation icon, as shown.

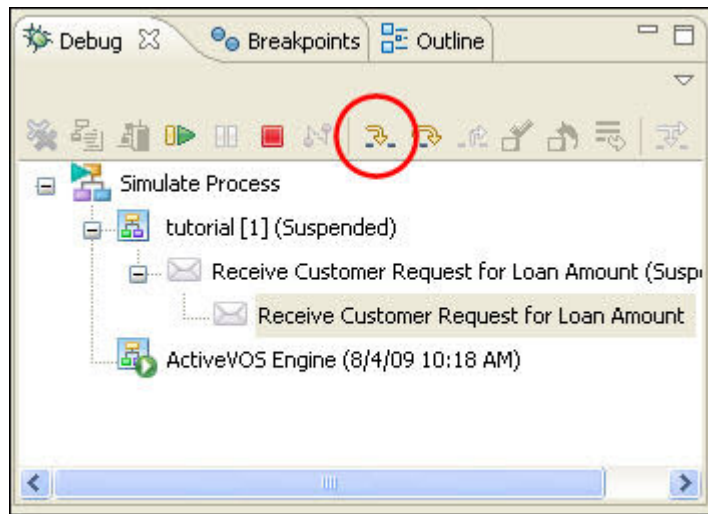


3. Notice the changes to the Process Developer environment, as shown in the following example.

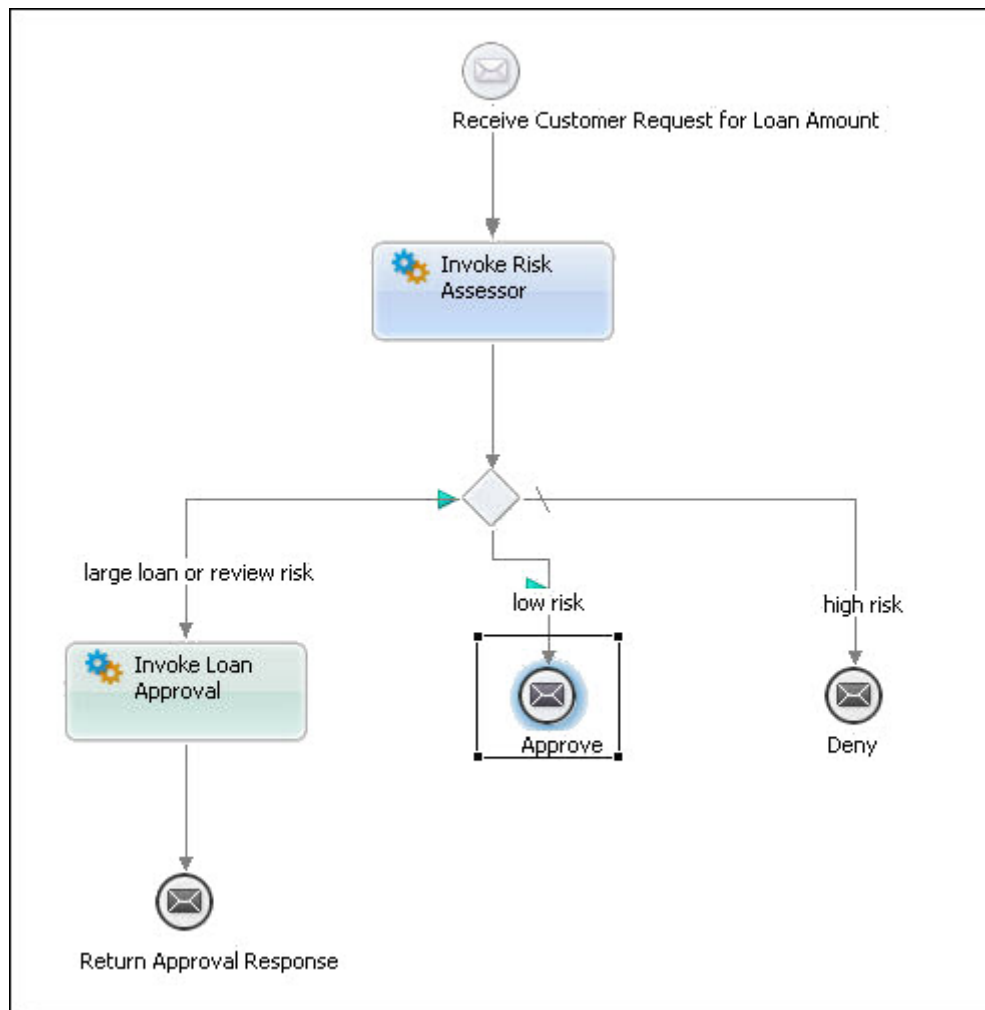


1	The perspective switches to a Debug perspective.
2	The beginning activity is highlighted
3	The current variable is highlighted
4	The sample data is cleared
5	The <i>Debug</i> view is opened to run and step through a process
6	The Process Console view is opened to report simulation events

4. In the Debug view, click the Step Into button to continue, as shown.



5. Step through your process, inspecting process variables as each activity is executed. The simulator stepped through the low risk path because Jones is a low risk and the loan amount requested was less than \$10,000. The Process Editor canvas should look like the following example when you are done.



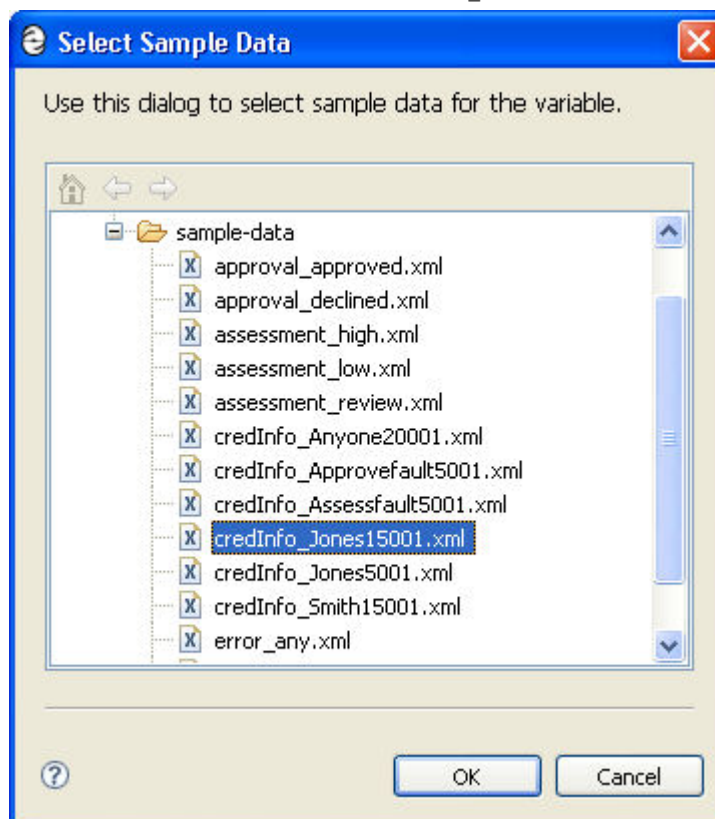
### Step 5: Clear the Process Execution State

1. Click anywhere on a blank part of the Process Editor canvas.
2. On the main toolbar, click the Clear Process Execution State icon, which is next to the **Start Simulation** icon.

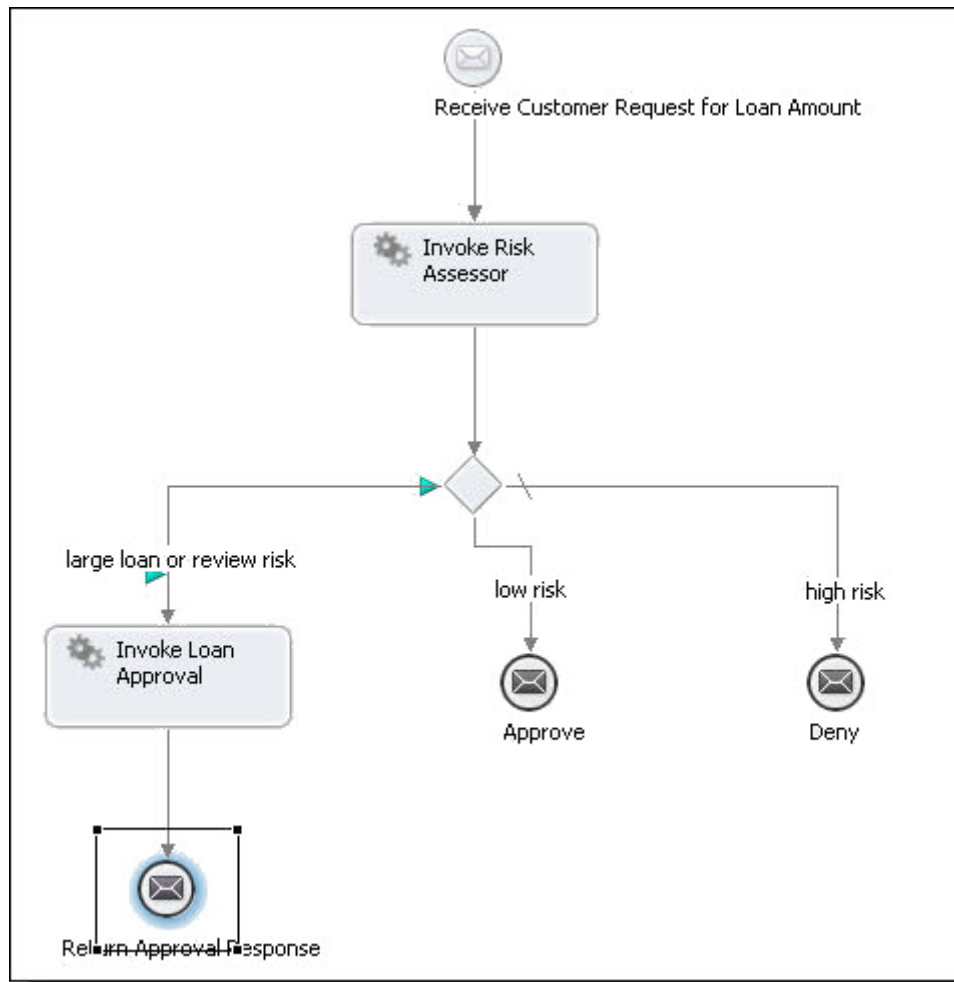
### Step 6: Override Default Values for Different Test Scenarios

The simulation path went through the risk assessor's service using the default sample data value of 5001. To test the loan approval path, do the following.

1. Select the Receive activity.
2. In the Process Variables view, open the `creditInformation` variable.
3. Right-mouse-click and select Load Data > from Project File.
4. From the sample-data folder, load `credInfo_Jones15001.xml`.



5. Save and rerun the process.
6. Notice that the simulation engine informs you that sample data is missing from the approval service, and presents a list of valid data files that match the message. Select `approval_approved.xml`.
7. Your simulation path should look like the following example:



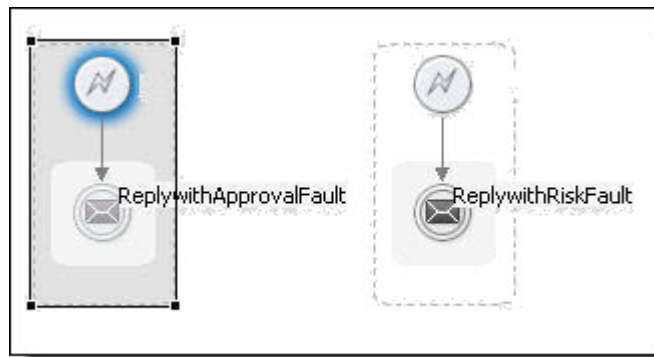
8. Clear the process execution state.
9. Inspect the Process Console view to see all simulation events.
10. From the Debug view toolbar, select Remove All Terminated Launches.

#### Step 7: Simulate Fault Handling

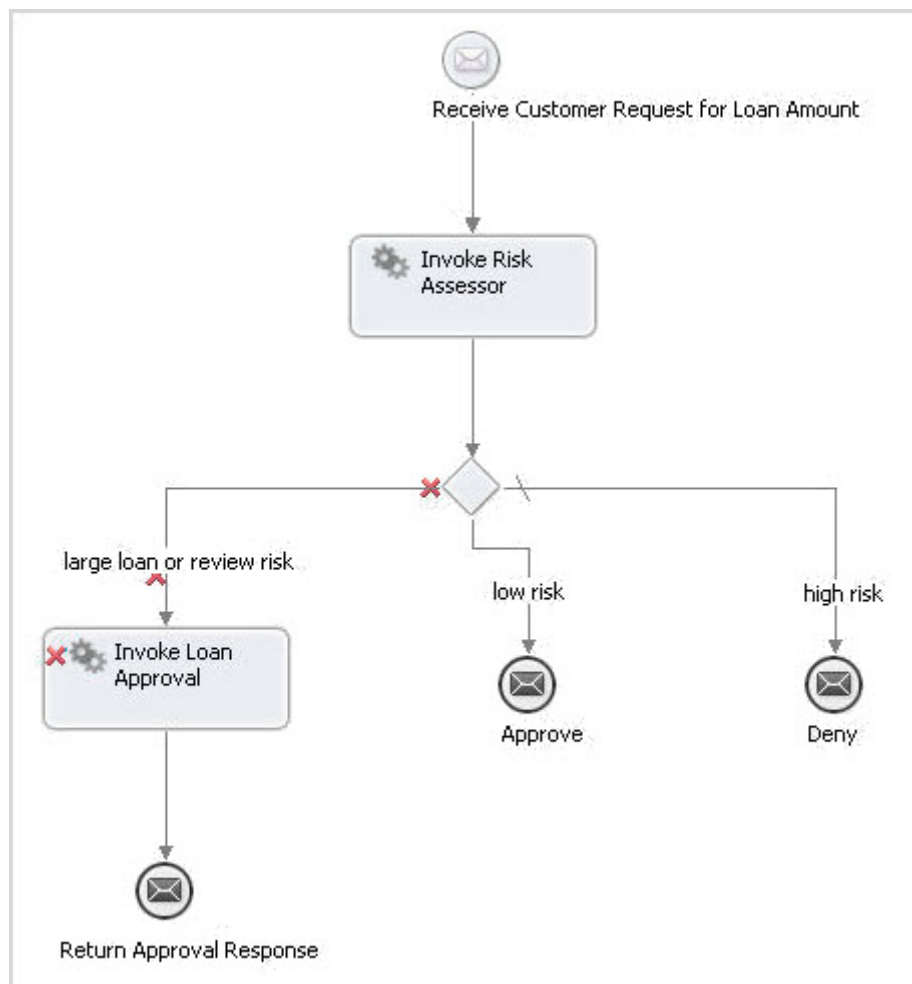
In the loan approval process, a fault is thrown if the loan approval or risk assessment service cannot handle the customer request. In ["Part 6: Adding Fault Handling" on page 64](#), you added two fault handlers to catch this fault and send a reply containing an error code. You will simulate this.

1. On the Process Editor canvas, select the Invoke Loan Approval activity.
2. In Properties view, select the All tab, and under Simulation, do the following:
  - a. Set Result to Fault.
  - b. Set Fault Name to loanProcessFault.
3. Start simulation and step through the process.
4. The simulator executes the fault handler, as shown.





5. In the Process Activities tab, the simulator shows the activity with a fault, as shown.



6. Look at the Process Console view to see execution path events.
7. Clear the process execution state.

**Tip:** You can also simulate fault handling for the risk assessment service. The same fault message is defined for both the loan approval and the risk assessment service.

To simulate fault handling for the risk assessment service, select the Invoke Risk Assessor activity and set the Result property to Fault and set the Fault Name. Change the sample data file to use an `amountRequested` to be less than 10000 and re-simulate.

## Part 9: Deploying the Process

If you have followed all tutorial parts so far, you have designed and debugged a BPEL process. You are now ready to deploy your process. Deployment is the act of publishing your BPEL process to the Process Server where it can run. In the deployment procedure, you use the WS-Addressing specification to define endpoint references.

It is time to deploy the process to the ActiveVOS embedded server. Deployment is the act of publishing your BPEL process to the ActiveVOS server where it can run. In the deployment procedure, you use the WS-Addressing specification to define endpoint references.

In the Project Explorer view of Process Developer, you should have the following files:

- `tutorial.bpel` that you created in Part 2
- `tutorialCompleted.bpel`, a completed version of the file created in Part 2 (optional)

After completing Part 9 of the tutorial, you will be able to:

- Complete the prerequisite checklist for deployment.
- Create a process deployment descriptor file.
- Create a business process deployment archive.
- Deploy the business process archive to the server.

### Step 1: Complete the Prerequisite Checklist for Deployment

A BPEL file is ready for deployment to the Process Server when there are no errors in the Problems view and when a simulated execution of the process terminates normally.

1. Open `tutorial.bpel`.
2. Simulate the process, as described in [“Part 8: Simulating the Process” on page 69](#), and test all execution paths.

### Step 2: Create a Process Deployment Descriptor File

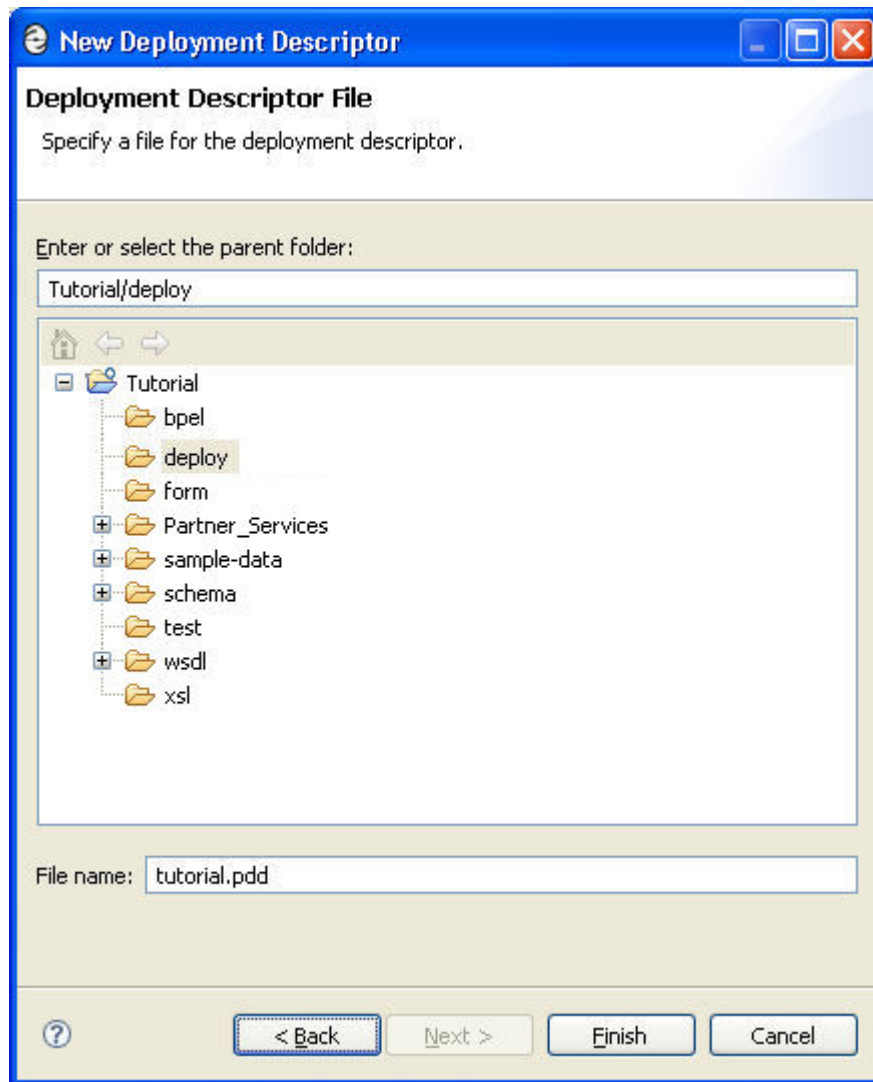
A Process Deployment Descriptor (`.pdd`) file describes the relationship between the participant partner links defined in the BPEL file and the implementation required to interact with actual partner endpoints. You create a `.pdd` file to add address information about your endpoint references. The `.pdd` file is an integral part of the deployment package for the process.

Recall that the loan approval process has the following definitions for partner links:

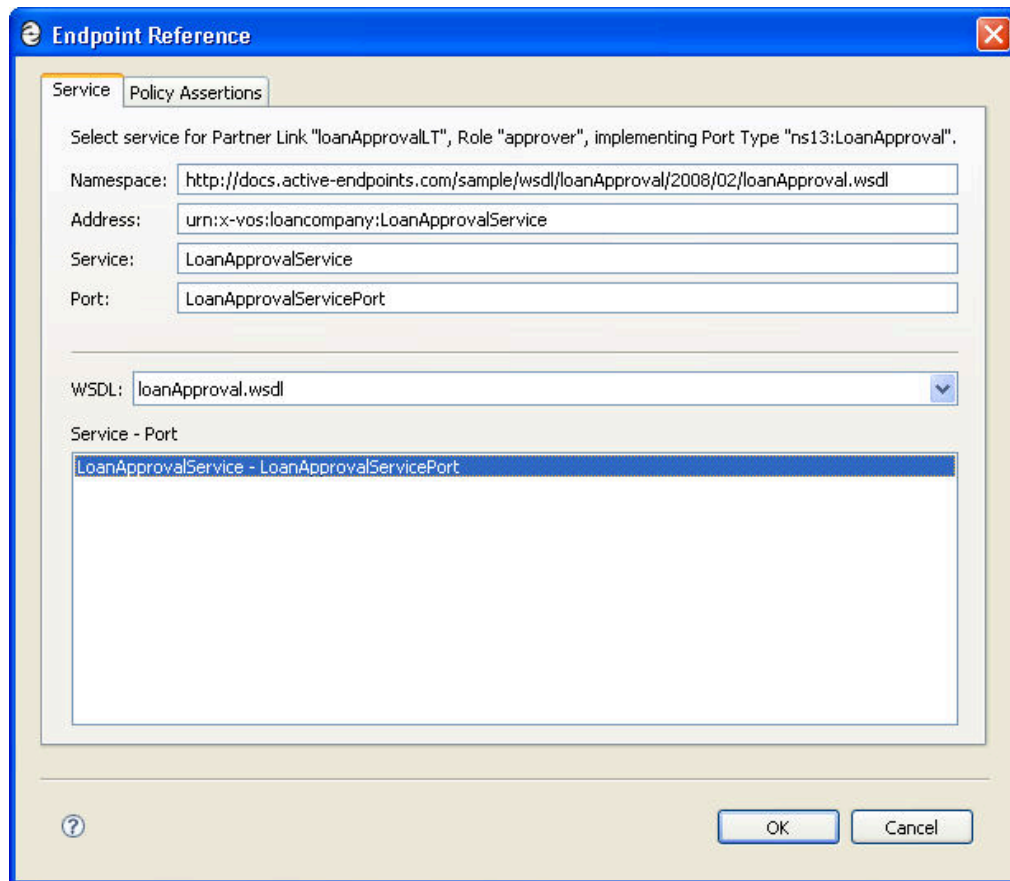
- `loanProcessor` is the process service consumer. In BPEL terminology, the My Role partner.
- `RiskAssessment` and `LoanApproval` are partner service providers. In BPEL terminology, the Partner Role partners.

You will assign an endpoint type for each partner role and will provide access protocol information for the process role (My Role). An endpoint type is a binding property that indicates the actual service the process interacts with. Different types give you control over specifying services you work with now and in the future. In this tutorial, you will indicate a static reference to partner services.

1. Select **File > New > Deployment Descriptor** to open the **New Deployment Descriptor** dialog.
2. Select `tutorial.bpel`, and click **Next**.
3. Select the `deploy` folder to store your deployment descriptor, shown in the example below, and click **Finish** to open the PDD Editor.



4. Do not make any changes on the General tab.
5. On the Partner Links tab, select the `LoanApproval` partner link.
6. Select WSA Address from the Invoke Handlers list. This addressing technique provides flexibility in deploying your processes into different server locations.
7. Select the dialog (...) button next to the Endpoint Reference text box. In the **Endpoint Reference** dialog, notice that `LoanApproval.wsdl` is selected, and the matching service is selected as shown. Select **OK**.




WS-Addressing is automatically added based on the binding information in the WSDL file.

8. On the Partner Links tab, leave Static as is from the Endpoint Type list, as shown in the example.

### Partner Links

RiskAssessment: Missing a partner role endpoint reference definition.

Name	Location	Status
LoanApproval		
loanProcessor		
 RiskAssessment		Missing a partner role endpoint reference definition.

#### Partner Role

Invoke Handler:

Endpoint type:

Endpoint Reference:

```
<wsa:EndpointReference xmlns:wsa="http://schema
<wsa:Address>urn:x-vos:loancompany:LoanAppro
<wsa:ServiceName PortName="LoanApprovalServic
</wsa:EndpointReference>
```

#### My Role

Binding:

Service:

Allowed Roles:

Policy:

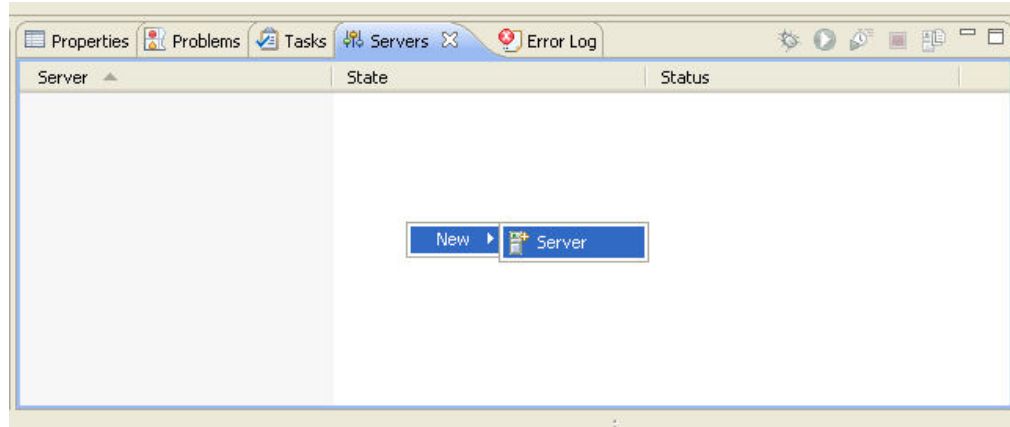
9. Select `RiskAssessment`, select `WSA Address` as the Invoke Handler, accept the `riskAssessment.wsdl` and service as the Endpoint Reference, and leave `Static` as the Endpoint Type.
10. Select `loanProcessor`, and in the `My Role` panel, select `Document Literal` for the Binding style.
11. In the `Service` field, type the name `TutorialService`.
12. Save and close `tutorial.pdd`.

### Step 3: Starting the Process Server

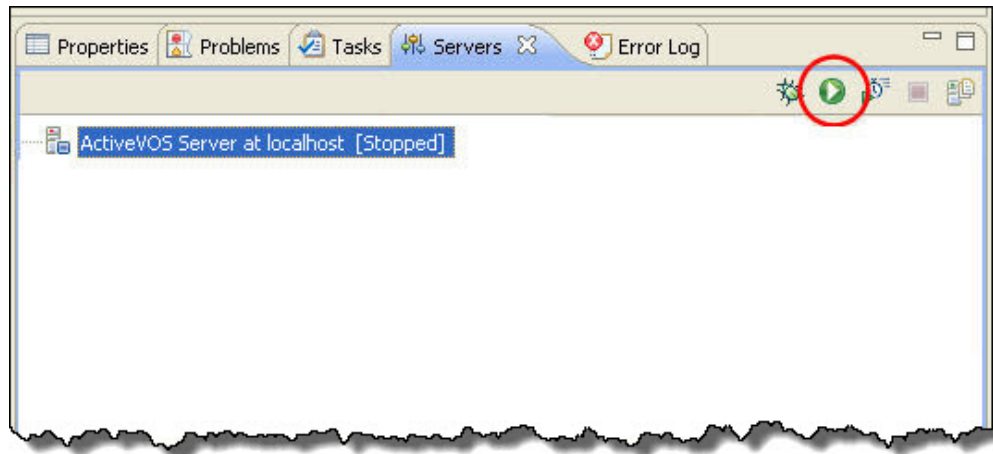
You will now deploy `tutorial.pdd` and its resources to the server. The first step is to start the server.

The Process Server consists of an engine running under Apache Tomcat. Tomcat is the servlet container that is used in the official Reference Implementation for the Java Servlet and JavaServer Pages technologies.

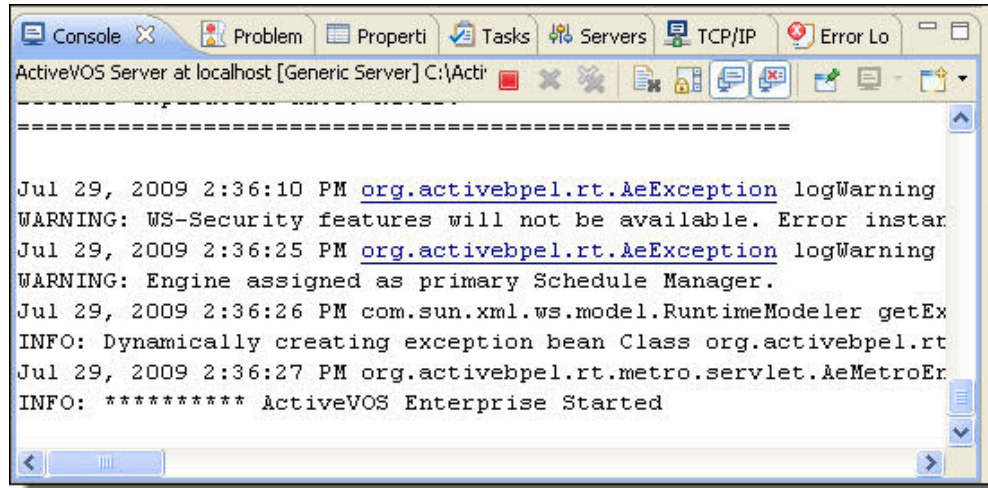
1. Select the Servers view in the lower right of the workspace and then right-mouse click within it. Select New > Server as shown in the illustration.



2. In the Server type list, select Process Server, click **Next**, and select Finish.
3. Select the Start the Server button, as shown in the example.



4. As the server starts up, you see start up tasks scroll in the Process Console. Files are deployed to the embedded server each time you start it. Many of these files are for BPEL for People activities that you may want to create for your next project. After the server starts, you'll see the message at the bottom of this figure.



#### Step 4: Create a Business Process Deployment Contribution Archive

To deploy your process to the Process Server, you must add all relevant files to an archive file (.bpr file), which is similar to a Web archive file.

1. Select **File > Export > Orchestration > Contribution-Business Process Archive File** and click **Next**.
2. Select the `tutorial.pdd` file to include in the archive, as shown in the following example.

**Export Business Process Archive**

**Contribution Specification**

Select deployment descriptor files to be contributed, location of the contribution (business process archive, BPR) and deployment method.

Select the deployment descriptors to contribute:

- ☒ Tutorial
  - ☒ deploy
    - ☒ tutorial.pdd

Select the export destination:

BPR file:

**Server Deployment Option**

Type:

Deployment URL:

Username:

Password:

**Options**

Group:

Description:

☒ Save the contribution specification as an Ant script in the workspace (.bprd)

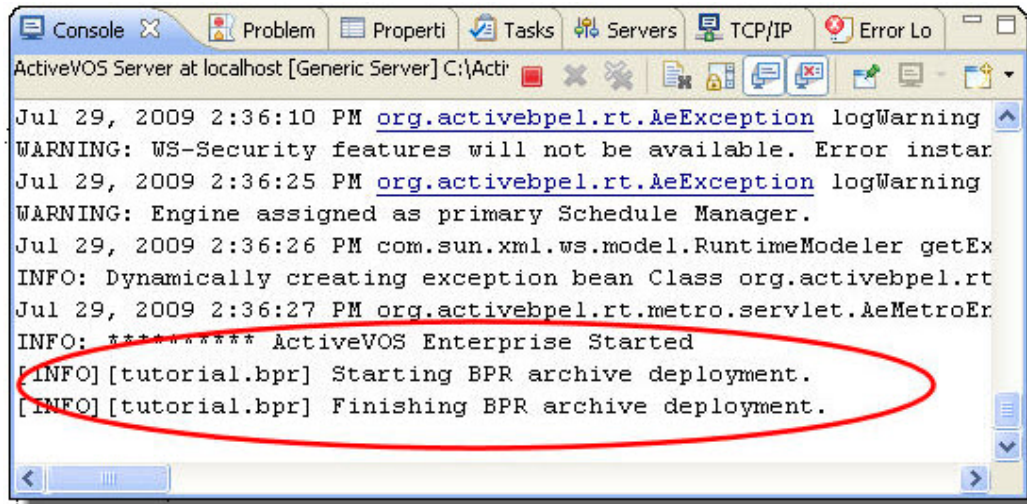
BPRD file:

- For the export destination BPR file, browse to the deploy folder and name the .bpr file `tutorial_completed.bpr`. Your path should be similar to the following:  
`Tutorial/deploy/tutorial_completed.bpr`
- In the Type field, select Web Service. The engine's default address is automatically filled in for the Deployment URL. If you changed your host and port information during installation, change the URL here.)  
Selecting this option automatically deploys your .bpr file to the Process Server upon completion of the export.
- Type in the Group name `Tutorials` and Description `Process Developer Tutorials`. These properties help you identify groups of processes on the server as you develop many types of processes.
- Select the check box next to `Save the contribution specification...`, and browse to the `deploy` folder.



7. Name the BPRD file `tutorial_completed.bprd`. This file is an Ant script which you can run to re-deploy the `bpr` file whenever you modify a `.bpel` or `.pdd` file.
8. Select Finish. A deployment details dialog shows the results.

Your BPR file has been automatically deployed to the server, as indicated by the information dialog. You can also see the results in the Process Console.



## Part 10: Creating a Form to Run the Process

To start at the beginning of the tutorial, see [Chapter 3, "ActiveVOS Tutorial" on page 34](#).

If you have followed all tutorial parts so far, you have deployed a BPEL process to Process Server.

After deploying the process to the Process Server, you must deploy the partner services, and then you can run the process by sending in a credit information request and getting back a response from the risk assessment or loan approval service.

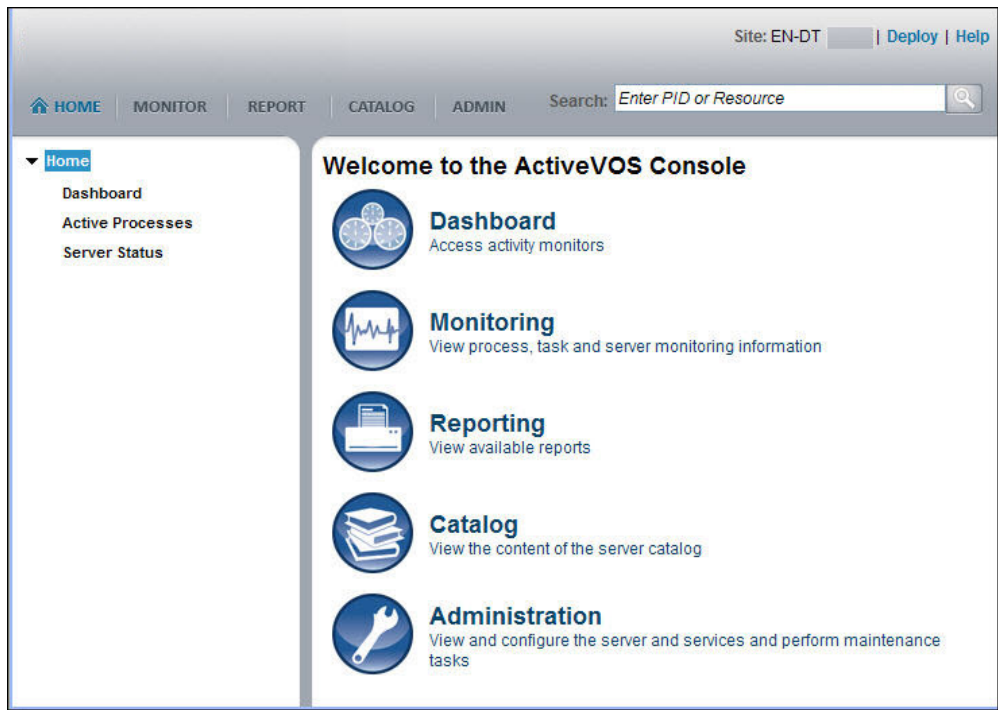
After completing Part 10 of the tutorial, you will be able to:

- View the Process Console in a workspace browser.
- Deploy the Risk Assessment and Loan Approval services.
- Create and deploy a process request form and Central Configuration file for Process Central.
- Sign in to Process Central and send a request to the process and receive a reply.

### Step 1: Open a Browser to View the Process Console

You can view deployment details for your processes in the Process Console.

1. On the Process Developer menu bar, select the Process Console icon.
2. The Process Console opens, using a URL of `http://localhost:8080/activevos`.



3. On the menu, select Admin.
4. Select URN Mappings. Notice that a URN to URL mapping is defined. This mapping is for the address of the Loan Approval and Risk Assessment services. (This URN mapping exists only in the embedded server in Process Developer. If you are deploying to a different server, such as JBoss, you will not see any pre-existing URN mappings. You must add them.)

### URN Mappings

<u>Delete</u>	<u>URN</u>	<u>URL</u>
<input type="checkbox"/>	urn:x-vos:loancompany	http://localhost:8080/active-bpel/services/\${urn.4}

---

**Add new URN Mapping**

URN

URL

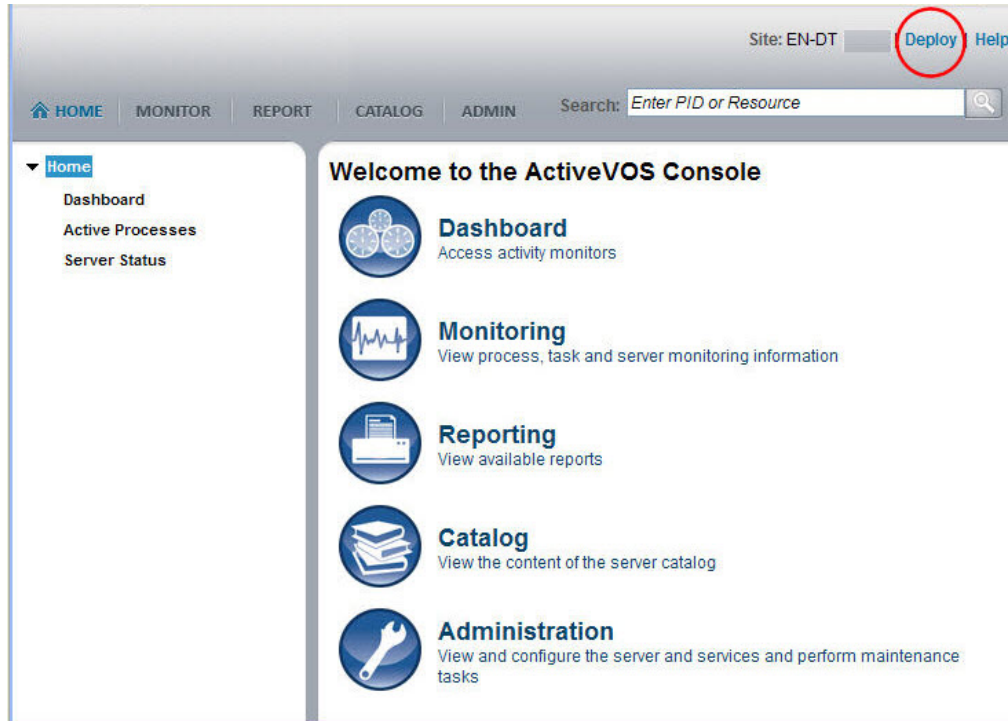
Normally you will add your own URN mappings. However, for the convenience of the tutorial, the partner role URNs were mapped to a URL for you.

**Note:** Be sure that the host port 8080 in the URL is correct. If you selected a different port during installation of Process Developer, select the URN to open the URL for editing. Update the URL so that the host port matches your server host port.

## Step 2: Deploy the Risk Assessment and Loan Approval Services

The Loan Approval and Risk Assessment services have been created as BPEL processes for the convenience of this tutorial, and they have already been packaged in a BPR file. You must deploy these services. Then, when you execute the tutorial process, one or both of these services will also execute. They will be invoked at the URL defined in the URN mapping.

1. In the Process Console, select Deploy as shown.



2. In the Deploy dialog, browse to the following location:

```
[Process Developer installation folder]\
Process Developer\workspace\Tutorial\
Partner_Services\riskAssessment.bpr
```

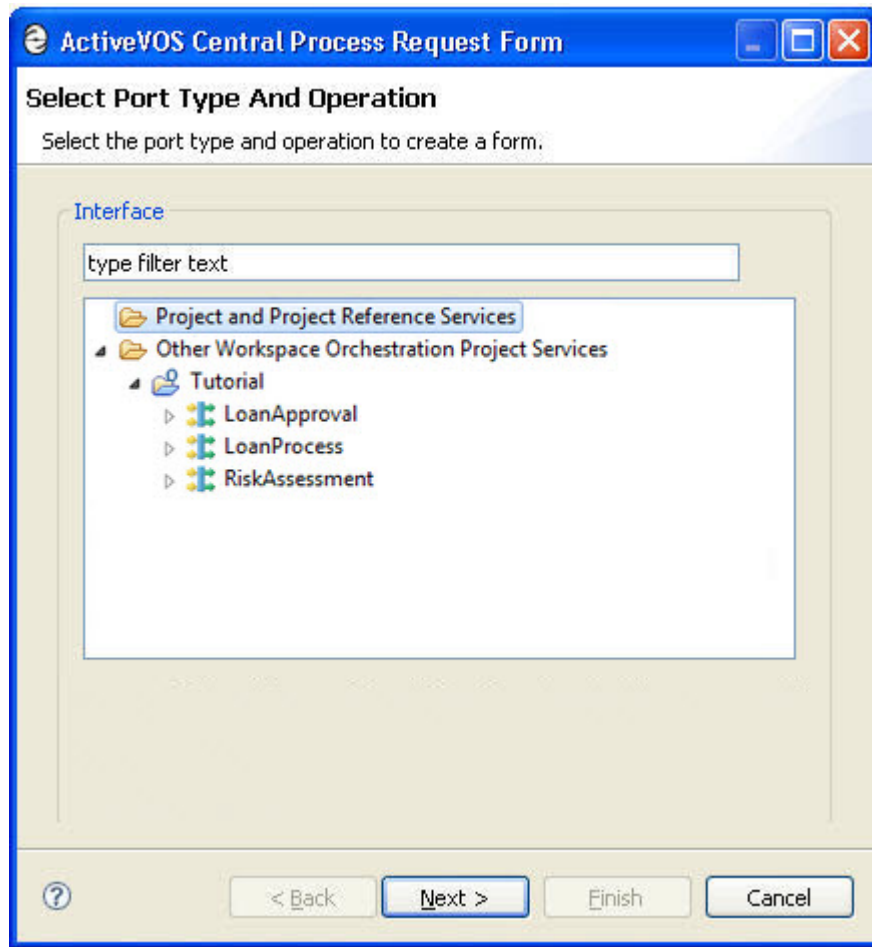
3. Deploy this BPR, and select Deploy another resource and select the `loanApproval.bpr`.

You can view the deployed processes. Do this in the Process Console by selecting **Catalog**, and then **Process Definitions**. You will see the listing for the tutorial, risk assessment, and loan approval processes.

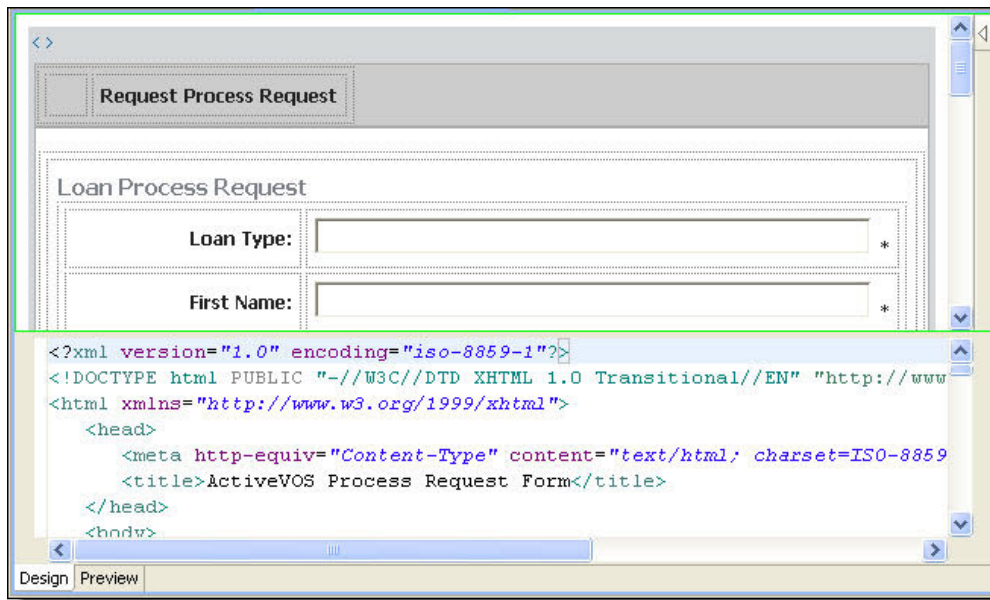
## Step 3: Create a Process Request Form

You will now use Process Central to send a request to the process to start it. Process Central is a client application that contains forms, as well as other work items, such as tasks and reports.

1. Select File > New > Process Request Form.
2. In the wizard, select the request operation, which is the operation for the starting receive activity.



3. Select Next and name the file `tutorialRequest.html`. Notice that the file will be added to a new folder called `request`. (This name is not related to the request operation. It is a generic name for all process requests that you may create for a project.)
4. Select Finish. Your process request form opens in the Web Page Editor, as shown.



5. Select the first occurrence of Request from the form title, *Request Process Request* and change it to Tutorial so that the title is Tutorial Process Request.
6. Notice that the input message is in a table, and the title of the table is Loan Process Request. This title is based on the schema element of the message. We will change it to provide instructions for testing different paths in the process.
7. Open the HTML Palette, and drop a table beside the input message header, as shown:



8. In code view, add the following HTML code between the `<table></table>` tags:

```
<tr><td>5000 &lt;loan amount &lt;=20000</td>
<td>Jones is declined</td></tr>
<tr><td>20000 &lt;loan amount &lt;=50000</td>
<td>Only Smith is approved</td></tr>
<tr><td>loan amount &gt; 50000</td>
<td>Everyone is declined</td></tr>
<tr><td>Last Name: Approvefault</td>
<td>Loan approval faults</td></tr>
<tr><td>Last Name: Assessfault</td>
<td>Risk Assessment faults</td></tr>
```

9. In code view, your HTML code should be:

```

<div>
  <span class="avc_heading">Loan Process Request</span>
  <table>
    <tr><td>5000 <lt; loan amount <lt;=20000</td><td>Jones is declined</td></tr>
    <tr><td>20000 <lt; loan amount <lt;=50000</td><td>Only Smith is approved</td></tr>
    <tr><td>loan amount <gt; 50000</td><td>Everyone is declined</td></tr>
    <tr><td>Last Name: Approvefault</td><td>Loan approval faults</td></tr>
    <tr><td>Last Name: Assessfault</td><td>Risk Assessment faults</td></tr>
  </table>
</div>

```

10. Your form should look like the following example:

Loan Process Request	
5000 <loan amount <=20000	Jones is declined
20000 <loan amount <=50000	Only Smith is approved
loan amount > 50000	Everyone is declined
Last Name: Approvefault	Loan approval faults
Last Name: Assessfault	Risk Assessment faults

Loan Process Request	
<b>Loan Type *</b>	<input type="text"/>
<b>First Name *</b>	<input type="text"/>
<b>Last Name *</b>	<input type="text"/>
<b>Day Phone *</b>	<input type="text"/>
<b>Night Phone *</b>	<input type="text"/>
<b>Social Security Number *</b>	<input type="text"/>
<b>Amount Requested *</b>	<input type="text"/>
<b>Loan Description *</b>	<input type="text"/>
<b>Other Info</b>	<input type="text"/>
<b>Response Email *</b>	<input type="text"/>
<b>First Approval Task Ref</b>	<input type="text"/>

11. Save and close your form.

#### Step 4: Create a Central Configuration File



To deploy your form to Process Central, you must create a configuration file that contains deployment information about the form. The purpose of the configuration file is similar to that of a Process Deployment Descriptor.

1. Select File > New > Central Configuration.
2. Select the deploy folder for the location.
3. Name the file tutorialRequest. The avccomfig extension is automatically added.
4. Select **Finish**, and the file opens in an XML Editor.
5. Select the Source tab to view the file.
6. Scroll down to the Requests section as shown.

```

<!-- Requests - example note that the values should be replaced with those from your
<tns:requestCategoryDefs>
  <tns:requestCategoryDef id="FILL_IN_CATEGORY_ID" name="FILL_IN_CATEGORY_NAME">
    <avccom:requestDef id="FILL_IN_REQUEST_ID" name="FILL_IN_REQUEST_NAME">
      <avccom:allowedRoles>
        <avccom:role>FILL_IN_ROLE_1</avccom:role>
        <avccom:role>FILL_IN_ROLE_2</avccom:role>
      </avccom:allowedRoles>
      <avccom:description>PROVIDE A REQUEST DESCRIPTION HERE.</avccom:description>
      <avccom:formLocation>project:/PATH/TO/REQUEST_FORM.HTML</avccom:formLocation>
    </avccom:requestDef>
  </tns:requestCategoryDef>
</tns:requestCategoryDefs>
-->

```

7. You need to uncomment the XML in this section. Add an HTML end comment tag (--) at the end of the Requests line, the top line shown in the illustration above.
8. Remove the end comment tag at the end of the Requests section.
9. Remove the <avccom:allowedRoles> section. There are four lines of code to remove. If you make a mistake, close the form, delete it from the deploy folder, and create a new form.
10. Fill in the placeholder values exactly as shown. Note that the values are case-sensitive:

```

<tns:requestCategoryDefs>
  <tns:requestCategoryDef id="education_category"
    name="Tutorial and Samples">
    <avccom:requestDef id="tutorial_request"
      name="Tutorial Request Form">
      <avccom:description>
        Submit loan approval request (basic tutorial).
      </avccom:description>
      <avccom:formLocation>
        project:/Tutorial/form/request/tutorialRequest.html
      </avccom:formLocation>
    </avccom:requestDef>
  </tns:requestCategoryDef>
</tns:requestCategoryDefs>

```

11. Save the file. Your completed file should look like this:

```

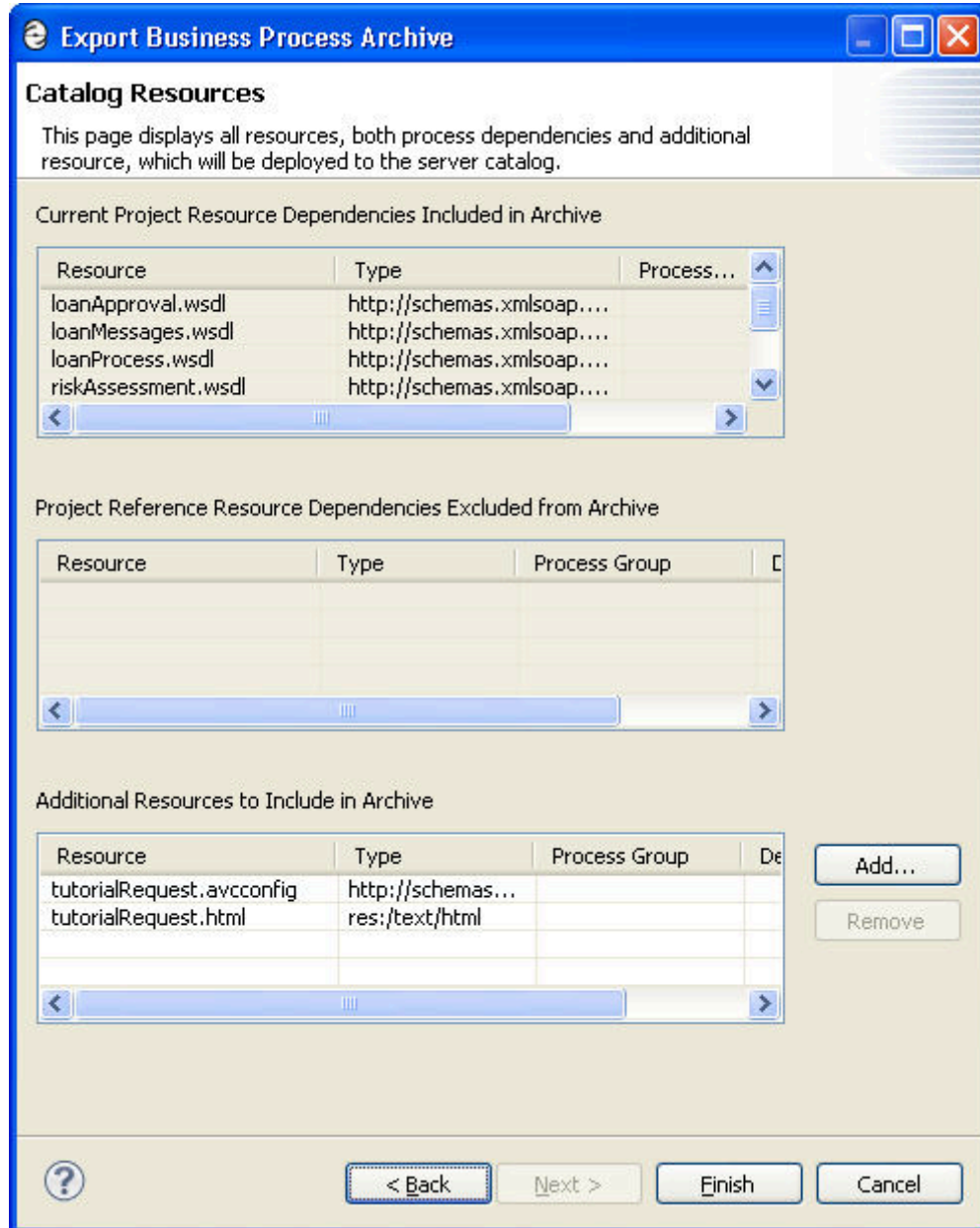
51 <!-- Requests -->
52 <tns:requestCategoryDefs>
53   <tns:requestCategoryDef id="education_category" name="Tutorial and Samples">
54     <avccom:requestDef id="tutorial_request" name="Tutorial Request Form">
55       <avccom:description>Submit loan approval request (basic tutorial).</avccom:description>
56       <avccom:formLocation>project:/Tutorial/form/request/tutorialRequest.html</avccom:formLocation>
57     </avccom:requestDef>
58   </tns:requestCategoryDef>
59 </tns:requestCategoryDefs>

```

## Step 5: Re-deploy Your BPR Contribution

Now that we have a request form and a configuration file describing how to display it in Process Central, you can deploy the BPR contribution, because the Tutorial project was updated with new files. Each time you update your project, redeploy the entire project to keep your files together as a contribution unit.

1. In the Project Explorer, select **File > Export > Contribution-Business Process Archive**.
2. Do not change any values on the first page of the wizard.
3. Select **Next**. Notice that two new resources were added to the contribution:



4. Select **Finish** to deploy `tutorial.bpr`. On the server, you now have two contributions. The older one (version 1) is offline and the new one (version 2) is online.

#### Step 6: Open Process Central and Submit a Request

Process Central is a great place to test the different paths of your process.

1. Ensure that the server is running and that you have deployed your BPR for your project.



2. You can open Process Central from the Process Developer toolbar to view an embedded browser. To open a larger, stand-alone window, select the Open Web Browser button from the toolbar.
3. In the Address bar, type in the following URL:  
http://localhost:8080/activevos-central
4. In the Sign On screen, sign on with the following user name and password:  
Username: manager  
Password: manager
5. In the navigation area underneath Home select Forms.
6. Select the Tutorial and Samples folder. This is the folder you created in the tutorialRequest.avcconfig file.
7. Click on the Tutorial Request Form in the work area. You will see the Tutorial Process Request form you created.
8. Submit a request as follows:
  - In the Last Name field, type in Jones (case-sensitive)
  - In the Amount Requested field, type in an amount between 5000 and 20000
  - Fill in the other fields as desired
9. Your form should look like the following:

### Loan Process Request

5000 <loan amount <=20000 Jones is declined  
 20000 <loan amount <=50000 Only Smith is approved  
 loan amount > 50000 Everyone is declined  
 Last Name: Approvefault Loan approval faults  
 Last Name: Assessfault Risk Assessment faults

Loan Type:  \*

First Name:  \*

Last Name:  \*

Day Phone:  \*

Night Phone:  \*

Social Security Number:  \*

Amount Requested:  \*

Loan Description:  \*

Other Info:

Response Email:  \*

First Approval Task Ref:

10. Scroll to the bottom of the form, and select Send Request.

11. The response from the loan approval process is displayed:

Tutorial Process Request

Process Output

Loan Approval Response

Response To Loan Request: declined

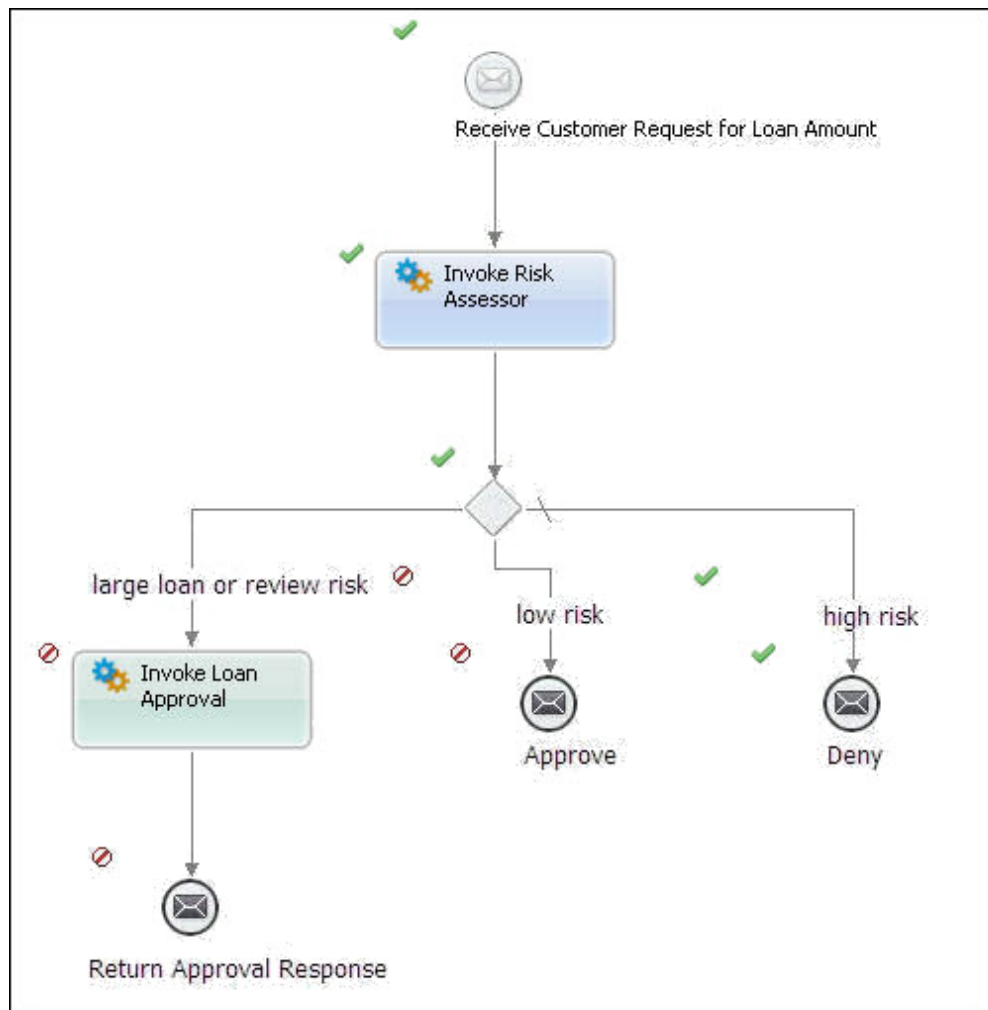
Response Description: We are sorry, this application falls outside of our credit risk guidelines.

Rejection Reason

Reason	Description
lowCredit	low credit score

### Step 7: View a Completed Process

1. In the Process Console, go back to the Service Definition page, and select Home from the menu, and then select Active Processes. Notice that two processes completed.
2. On the Active Processes page, select the tutorial process. You can see the actual results returned from the risk assessment service. Because Jones is a credit risk, the loan was denied.



For more details about the two services, and to debug your running or completed process from within Process Developer, go to [“Part 11: Debugging Your Process Remotely” on page 99](#).

## Part 11: Debugging Your Process Remotely

To start at the beginning of the tutorial, see [Chapter 3, “ActiveVOS Tutorial” on page 34](#).

If you have followed all tutorial parts so far, you have deployed and run a BPEL process on the Process Server.

Before starting this part of the tutorial, we recommend that you complete [“Part 10: Creating a Form to Run the Process” on page 89](#).

In the Project Explorer view of Process Developer, you should have the following files:

- `tutorial.bpel` that you created in Part 4
- `tutorial.bpr` that you created in Part 9

After completing Part 11 of the tutorial, you will be able to:

- Add a breakpoint to `tutorial.bpel`.
- Create a configuration file for launching a remote debugging session.
- Start a remote debugging session.
- Step through your process and inspect variables.

### Step 1: Add a Breakpoint to `tutorial.bpel`

You can connect to a running or completed process from within Process Developer for remote debugging. There are several options for remote debugging, and we will set a breakpoint in the process and then attach to the running process when the breakpoint is hit.

1. Open `tutorial.bpel` in the Process Editor.
2. Right-mouse click on the Receive activity and select Add Breakpoint. The Receive activity should look like the following example.



### Step 2: Start the server

The server must be running in order to start remote debugging, so start the server now (if it is not already running) before setting up a debug configuration.

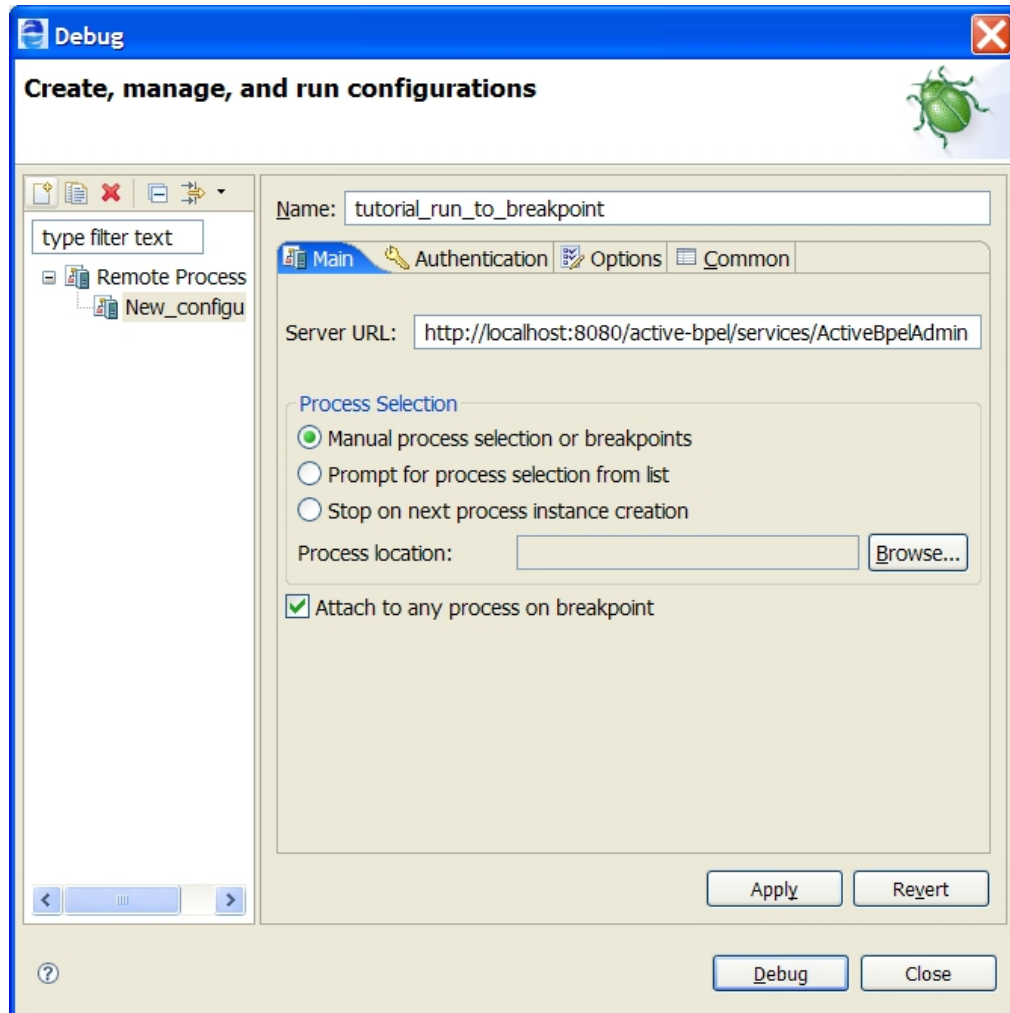
1. If the server is already running, as described in Part 10 of the tutorial, you can skip to Step 3.
2. Select the Servers view in the lower right of the workspace, and select Start the Server.

### Step 3: Create a Configuration for Launching a Remote Debug Session

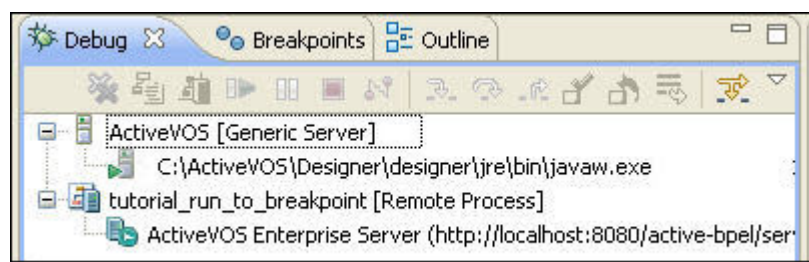
A remote debug configuration indicates where processes are running and how you want to attach to them.

1. From the Run menu, select Debug Configurations.
2. In the **Debug** dialog, select Remote Process and then select the New Launch Configuration icon in the toolbar.
3. In the Name field, type `tutorial_run_to_breakpoint`.
4. In the Main tab, note that the default Server URL is displayed for deployed server processes.

5. In the Process Selection panel, select Manual process selection or breakpoints. This selection indicates that you do not want to immediately debug a process, but rather will select a process manually.
  6. Enable the option Attach to any process on breakpoint to indicate that a process containing a breakpoint will be attached to, if not already attached, when a breakpoint is hit.
- Your **Debug Create, manage, and run configurations** dialog should look like the following example.



7. Select Debug to begin remote debugging for the current configuration.
- Your Debug view should look like the following example.



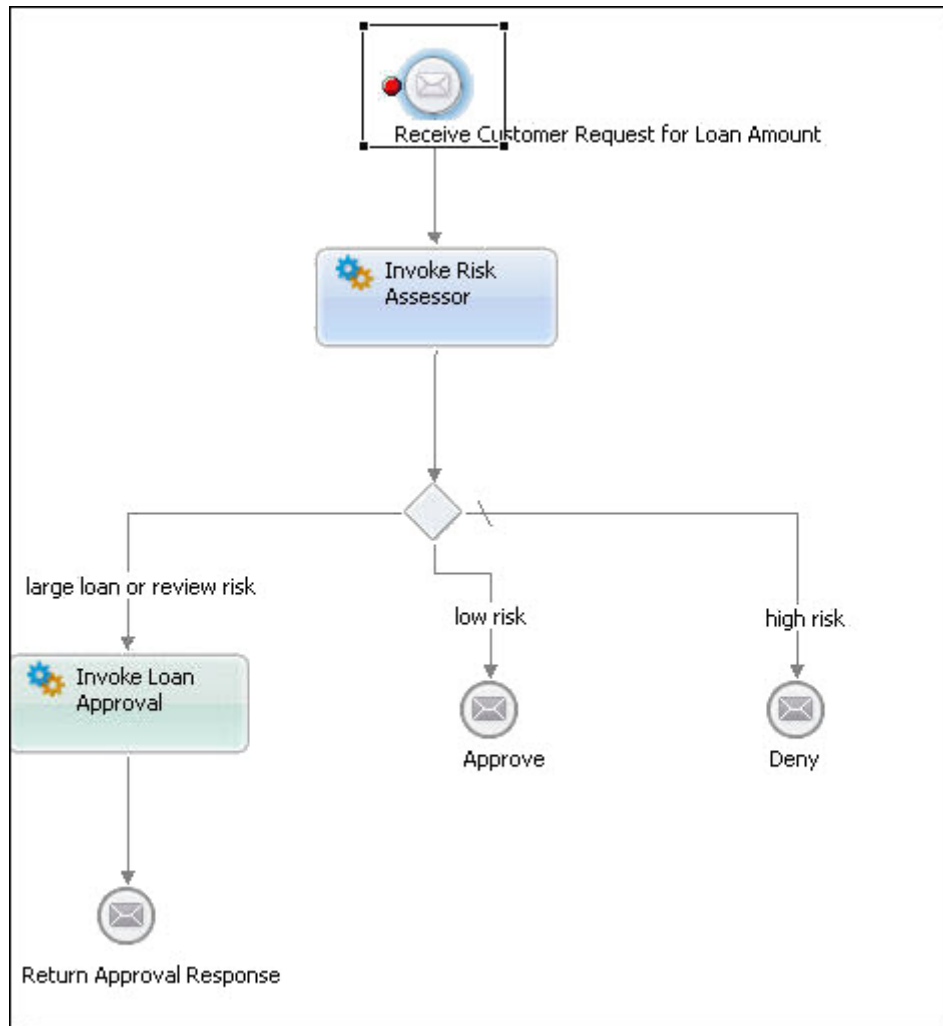
#### Step 4: Instantiate the Process

You can create an instance of the process by using the Process Central request form, as you did in Part 10 of the tutorial.

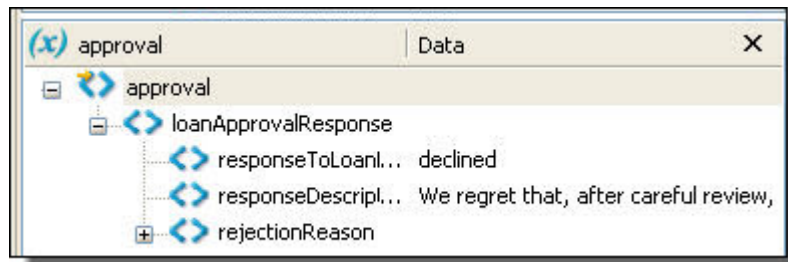
1. In Process Central, fill in the form with Last Name of `Jones` and amount-Requested value of `10000`.
2. Select Send Request.

#### Step 5: Begin Remote Debugging

1. In Process Developer, open all variables in the Process Variables view and position the view next to the Process Editor canvas. Your Process Editor should look like the following example.



2. Notice that the running process is stopped on the Receive activity with the breakpoint, as we configured.
3. In Debug view, click the Step Into icon on the toolbar. The debug highlighter moves to the Invoke Risk Assessor activity. The `creditInformation` variable displays the data submitted by the Process Central request.
4. Click Step Into twice to terminate and disconnect the process automatically.
5. Inspect the `approval` variable. Notice that Jones is not going to get a loan. The reply indicates a decline, as shown.



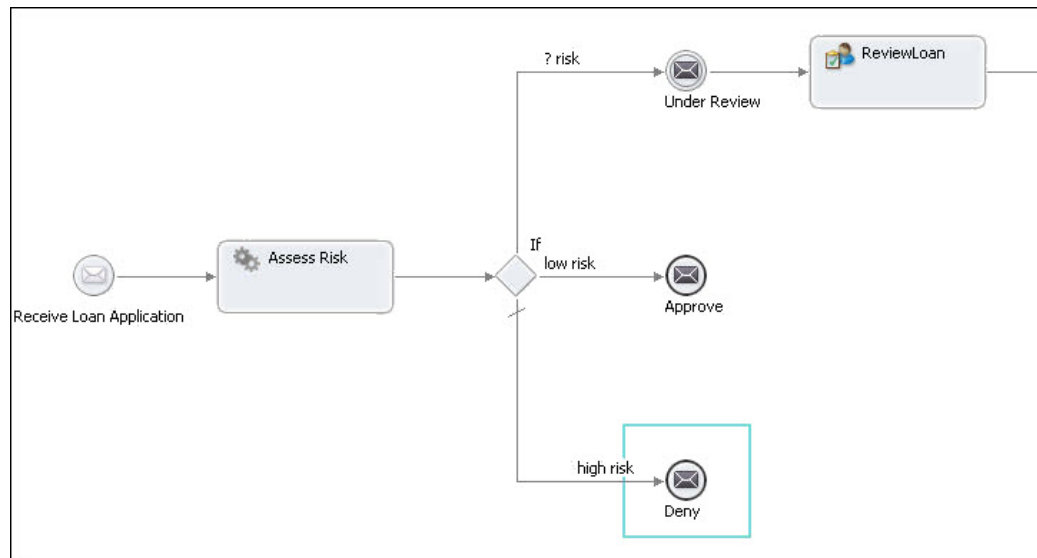
6. Repeat the above steps by adding different *Last Name* and *amountRequested* values.

The LoanApproval and RiskAssessment services are built on the following logic:

5000 <loan amount <=20000	Jones is declined, all others approved
20000<loan amount<=50000	Only Smith is approved, all others declined
loan amount>50000	Everyone is declined
Last name: Approvefault	Loan approver faults
Last name: Assessfault	Loan assessor faults

### Next Steps

Create a new orchestration project for Human Approval Completed. This process replaces the Invoke Loan Approval activity with a People activity, as shown in the illustration. The documentation that accompanies the Human Task (BPEL for People) sample describes how to deploy and run the sample. You can also review the Human Tasks help for details on building a People activity to add human workflow to your BPEL process.



## Part 12: Using the Web Services Explorer to Start a Process

If you deployed your tutorial to Process Server and do not yet have an Identity Service enabled, you will not be able to sign into Process Central. Process Central requires that authenticated users sign in, enabled through

the Identity Service in Process Server. The Process Central embedded in Process Developer is already enabled.

To start your process from an Process Server, such as JBoss or WebLogic, use the following instructions to use the Web Services Explorer.

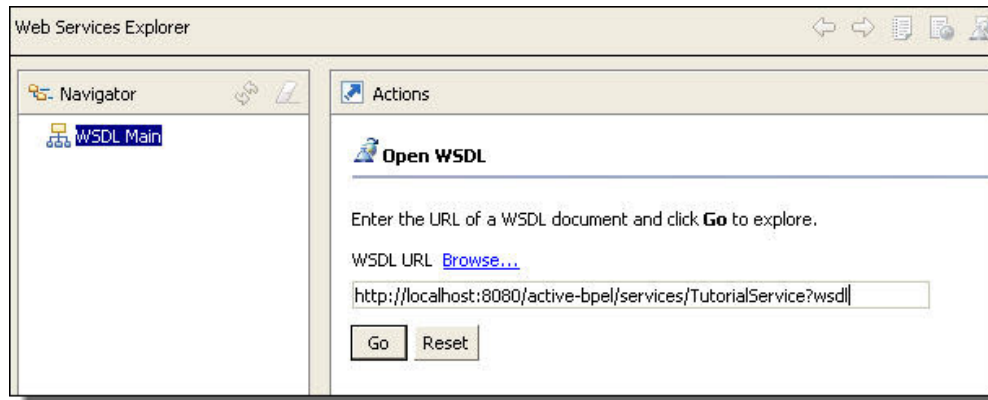
### Step 1: Launch the Web Services Explorer

You need to copy the `TutorialService` URL into the Web Services explorer in order to send a request to the process.

1. From the Process Developer Run menu, select Launch the Web Services Explorer.
2. Select the icon (upper right pane) for WSDL Page.
3. Select WSDL Main.
4. Return to the browser, and in the Process Console, select Service Definitions.
5. On the **Service Definitions** page, select `TutorialService`.
6. From the address bar, copy the `TutorialService` URL:

`http://localhost:8080/active-bpel/services/TutorialService?wsdl`

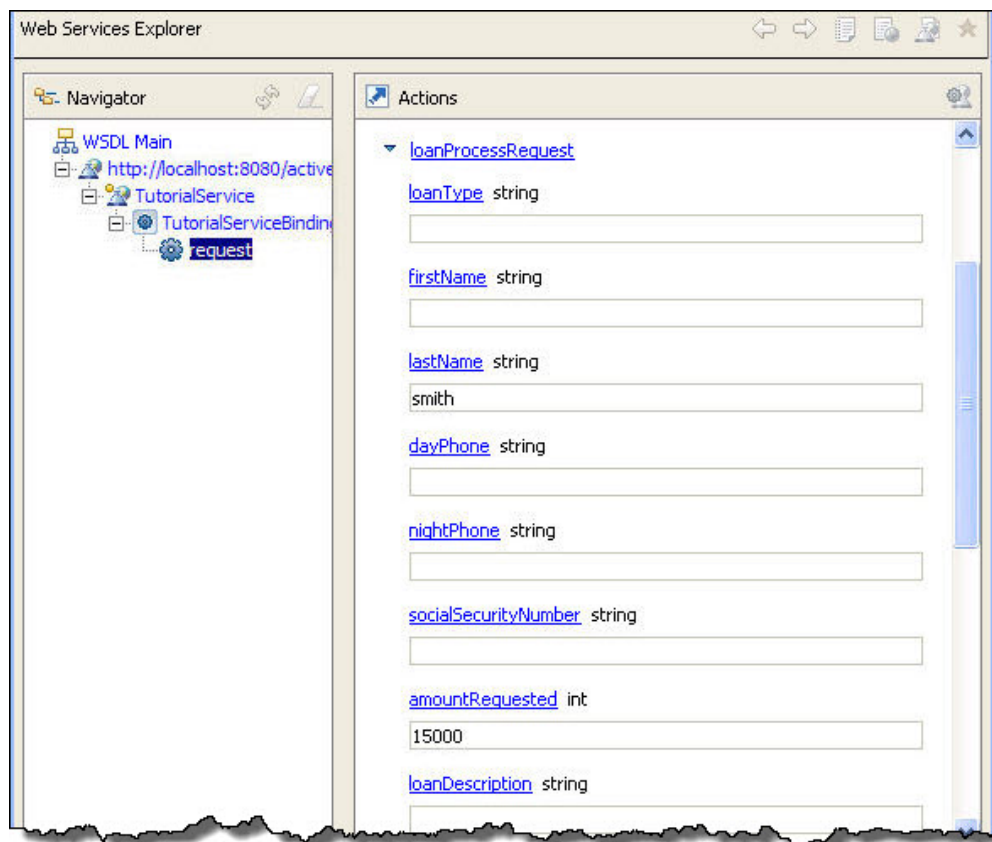
7. Paste this URL into the Web Services Explorer WSDL URL field as shown:



### Step 2: Send a Request to the Process

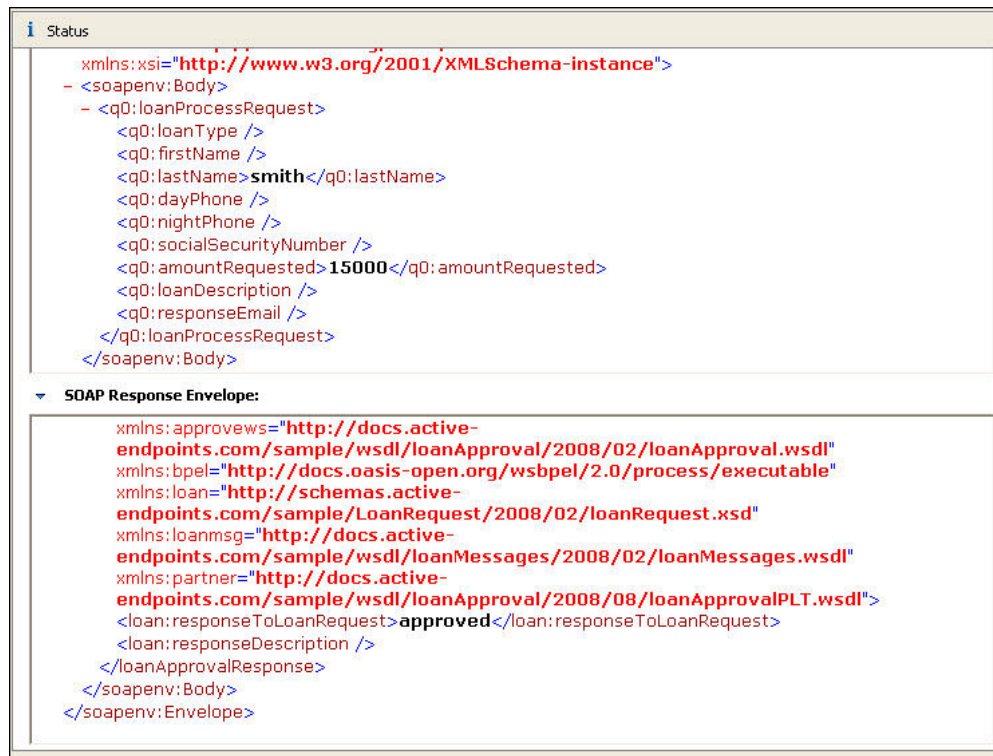
You can create an instance of the process by using the Web Services Explorer to initiate the Receive activity in `tutorial.bpel`.

1. In the Web Services Explorer, select Go under the WSDL URL field.
2. In the Bindings table, select the link for the supported SOAP binding (SOAP 1.1).
3. In the Operations table, select the request link.
4. Fill in the message parts. The required parts are Last Name and amountRequested. Fill in `Smith` for Last name and a high amount, such as `15000`, as shown:



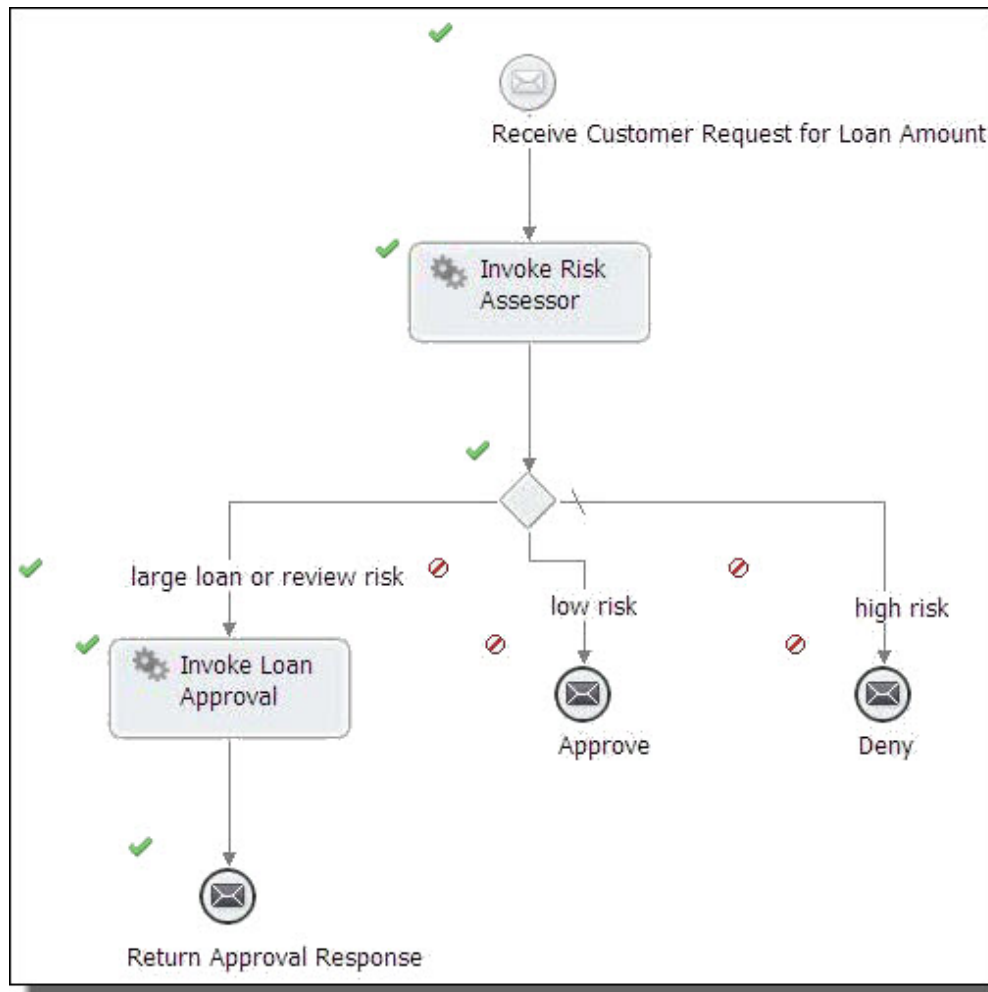
5. Select Go, which is located at the bottom of the message.
6. Click the Source link, and then double-click the Status box to see the SOAP request and response sent and received.





7. In the Process Console, go back to the Service Definition page, and select Home from the menu, and then select Active Processes. Notice that two processes completed.

8. On the **Active Processes** page, select the tutorial process. You can see the actual results returned from the loan approval service. The high amount loan request was sent to the approver. The approver approved the loan.



For more details about the two services, and to debug your running or completed process from within Process Developer, go to [“Part 11: Debugging Your Process Remotely” on page 99](#).