TIBCO ActiveMatrix BusinessWorks™ BPEL Extension

User's Guide

Software Release 5.9
November 2010

Important Information

Contents

Figures

Tables

Preface

This software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. Please see the readme.txt file for the availability of this software version on a specific operating system platform.

TIBCO ActiveMatrix BusinessWorks BPEL Extension allows you to create WS-BPEL orchestration processes within the TIBCO ActiveMatrix BusinessWorks framework. This product builds upon the strength of TIBCO ActiveMatrix BusinessWorks for defining the execution of automated business processing and adds the capability to define service-based orchestrations. WS-BPEL defines the interactions between the services, but it is not involved in the details of the implementation of each service. TIBCO ActiveMatrix BusinessWorks and the ActiveMatrix BusinessWorks BPEL Extension together provide a powerful application development platform that allows you to define a service-oriented architecture and develop the implementation of services.

Topics

Changes from the Previous Release of this Guide

This section itemizes the major changes from the previous release of this guide.

TIBCO ActiveMatrix BusinessWorks 5.9 support

TIBCO ActiveMatrix BusinessWorks BPEL Extension supports TIBCO ActiveMatrix BusinessWorks 5.9 release.

This release of TIBCO ActiveMatrix BusinessWorks BPEL Extension does not support creation and running of deployment artifacts (DAA) in the TIBCO ActiveMatrix environment.

Related Documentation

This section lists documentation resources you may find useful.

TIBCO ActiveMatrix BusinessWorks BPEL Extension Documentation

The following documents form the TIBCO ActiveMatrix BusinessWorks BPEL Extension documentation set:

- *TIBCO ActiveMatrix BusinessWorks BPEL Extension User's Guide* This manual outlines the features and functionality of the TIBCO BusinessWorks BPEL Extension. Knowledge of TIBCO BusinessWorks, Web Services, and BPEL is required before reading this manual.
- *TIBCO ActiveMatrix BusinessWorks BPEL Extension Release Notes* Read the release notes for a list of known issues for this release.

Other TIBCO Product Documentation

You may find it useful to read the documentation for the following TIBCO products:

- TIBCO ActiveMatrix BusinessWorks™ software: This software is a standards-based, easy-to-deploy solution for companies looking to integrate their enterprise computing environment and automate their business processes. You should be familiar with TIBCO ActiveMatrix BusinessWorks features and functionality, such as projects, mapping and transformation, WSDL files, and so on, before attempting to use the TIBCO ActiveMatrix BusinessWorks BPEL Extension.
- TIBCO Runtime Agent™ software: This software suite is a prerequisite for other TIBCO software products. In addition to TIBCO Runtime Agent components, the software suite includes the third-party libraries used by other TIBCO products, TIBCO Designer™, Java Runtime Environment (JRE), TIBCO Hawk® Agent, and TIBCO Rendezvous®.
- TIBCO Administrator™ Enterprise Edition software: This software allows you to manage users, machines and applications defined in a TIBCO administration domain. The TIBCO Administrator graphical user interface enables users to deploy, monitor, and start and stop TIBCO applications.
- TIBCO Designer™ software: This graphical user interface is used for designing and creating integration project configurations and building an Enterprise Archive (EAR) for the project. The EAR can then be used by TIBCO Administrator for deploying and running the application.

- TIBCO Enterprise Message Service™ software: This software lets application programs send and receive messages using the Java Message Service (JMS) protocol. It also integrates with TIBCO Rendezvous and TIBCO SmartSockets™ messaging products.

Third-Party Documentation

- The current draft of the BPEL 2.0 specifications can be found at http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v20-rddl.html.

Typographical Conventions

The following typographical conventions are used in this manual.

Table 1 General Typographical Conventions

Convention	Use
TIBCO_HOME	Many TIBCO products must be installed within the same home directory. This directory is referenced in documentation as *TIBCO_HOME*. The value of *TIBCO_HOME* depends on the operating system. For example, on Windows systems, the default value is `C:\tibco`.
`code font`	Code font identifies commands, code examples, filenames, pathnames, and output displayed in a command window. For example: Use `MyCommand` to start the foo process.
`bold code font`	Bold code font is used in the following ways: • In procedures, to indicate what a user types. For example: Type the username **`admin`**. • In large code samples, to indicate the parts of the sample that are of particular interest. • In command syntax, to indicate the default value.
italic font	Italic font is used in the following ways: • To indicate a document title. For example: See *TIBCO BusinessWorks Concepts* for more details. • To introduce new terms. For example: A portal page may contain several *portlets*. Portlets are mini-applications that run in a portal. • To indicate a variable in a command or code syntax that you must replace. For example: `MyCommand` *pathname*
Key combinations	Key names separated by a plus sign indicate keys pressed simultaneously. For example: Ctrl+C. Key names separated by a comma and space indicate keys pressed one after the other. For example: Esc, Ctrl+Q.

How to Contact TIBCO Support

For comments or problems with this manual or the software it addresses, please contact TIBCO Support as follows.

- For an overview of TIBCO Support, and information about getting started with TIBCO Support, visit this site:

  http://www.tibco.com/services/support

- If you already have a valid maintenance or support contract, visit this site:

  https://support.tibco.com

  Entry to this site requires a user name and password. If you do not have a user name, you can request one.

Chapter 1 Overview of TIBCO ActiveMatrix BusinessWorks BPEL Extension

This chapter describes the basic concepts of the TIBCO ActiveMatrix BusinessWorks BPEL Extension product.

Topics

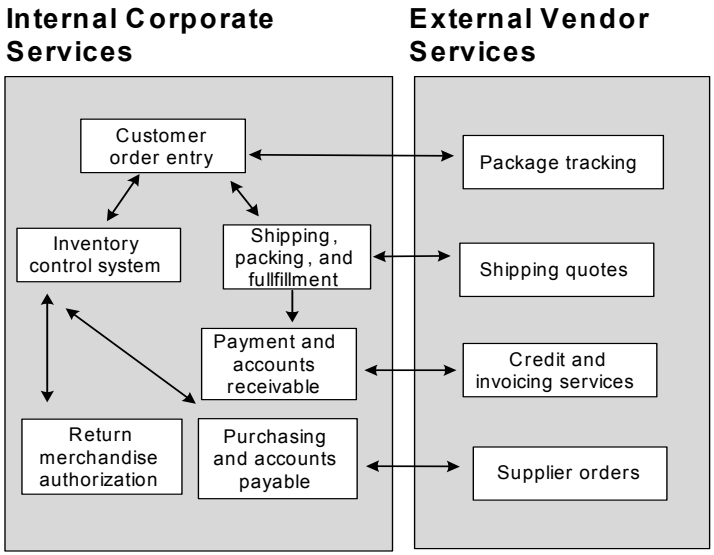
Product Overview

Web Service Business Process Execution Language, or WS-BPEL, is a standards-based language for defining how business processes interact. WS-BPEL processes describe the orchestration, or large scale interaction, of services within and outside of an enterprise computing environment. WS-BPEL builds upon several standards, such as XML, Web Service Description Language (WSDL), and XPath.

WS-BPEL is an important part of a service-oriented architecture (SOA). SOA is a software architecture in which applications and data are decomposed into discrete, independent services. Decomposing applications into services allows enterprise application components to be reused and integrated in flexible and efficient ways. Enterprise applications are more agile when using SOA because the components can switch to use new services as their needs dictate.

Figure 1 illustrates a simple business service model. The model describes a business that accepts orders from customers and fulfills those orders by obtaining the requested items from various suppliers. The business also interacts with shipping vendors for delivering the orders to the customer.

Figure 1 Example of business service model



TIBCO ActiveMatrix BusinessWorks BPEL Extension allows you to create WS-BPEL orchestration processes within the TIBCO ActiveMatrix BusinessWorks framework. This product builds upon the strength of TIBCO ActiveMatrix BusinessWorks for defining the execution of automated business processing and

adds the capability to define service-based orchestrations. WS-BPEL defines the interactions between the services, but it is not involved in the details of the implementation of each service. TIBCO ActiveMatrix BusinessWorks and the ActiveMatrix BusinessWorks BPEL Extension together provide a powerful application development platform that allows you to define a service-oriented architecture and develop the implementation of services.

Prerequisite Information

WS-BPEL is built upon many standards, and many features of the TIBCO ActiveMatrix BusinessWorks BPEL Extension rely on functionality provided within TIBCO ActiveMatrix BusinessWorks. Before using TIBCO ActiveMatrix BusinessWorks BPEL Extension, it would be helpful to understand the following:

- TIBCO ActiveMatrix BusinessWorks — the ActiveMatrix BusinessWorks BPEL Extension uses the same process design and administration interfaces as TIBCO ActiveMatrix BusinessWorks. You should understand TIBCO ActiveMatrix BusinessWorks before starting to use the ActiveMatrix BusinessWorks BPEL Extension.
- Simple Object Access Protocol (SOAP) and Web Service Description Language (WSDL) — web services are the underlying model for WS-BPEL. SOAP and WSDL are the foundations for web service communication.
- XML Schema, XPath, and XSLT — Data is described in XML schemas, and transformation and manipulation of data are performed through XPath and XSLT.

You should have a solid understanding of these concepts before using TIBCO ActiveMatrix BusinessWorks BPEL Extension.

Services

Services are the building blocks of a service-oriented architecture. Services provide an abstraction of the applications and interactions that make up a business process.

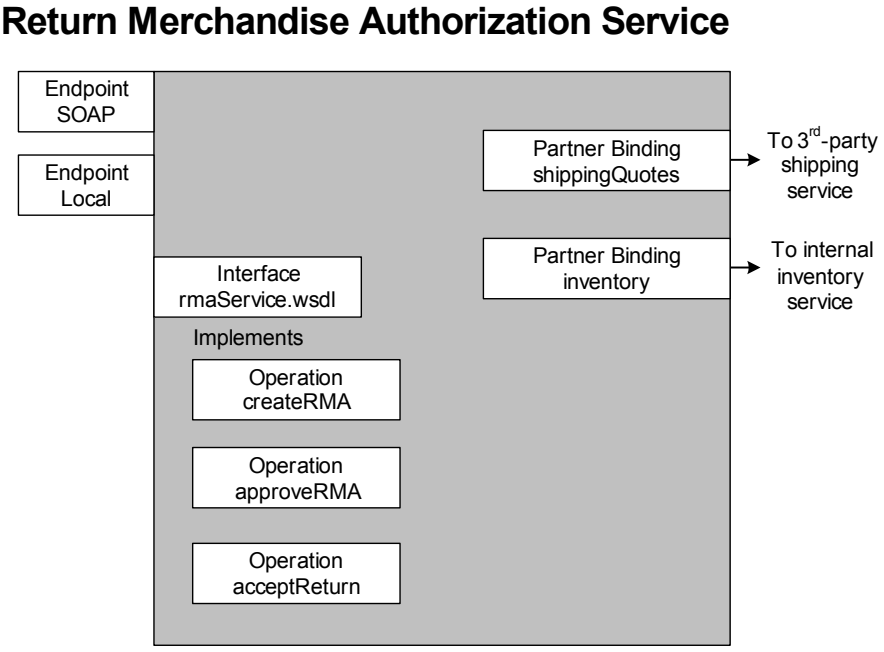
Services in TIBCO ActiveMatrix BusinessWorks are similar to services in the ActiveMatrix BusinessWorks BPEL Extension, and the same Service resource is used to define either type of service. However, TIBCO ActiveMatrix BusinessWorks services do not have the same configuration options or parts as Services within the ActiveMatrix BusinessWorks BPEL Extension.

This section describes Services within the TIBCO ActiveMatrix BusinessWorks BPEL Extension.

For example, in your business, you may have a return merchandise authorization process to allow customers to return damaged or unwanted products. Your process may involve gathering the original purchase order information, determining whether the merchandise can be returned, offering the shipping costs for return, and accepting the returned merchandise when it arrives. This process is a complex set of interactions between several people, internal applications, and third-party services.

Defining a service for the return merchandise authorization allows you to view your business processes from a higher level and orchestrate the interaction of this service with other services, such as inventory or a third-party shipping quotes service. Figure 2 illustrates the parts of a service using the return merchandise authorization service example.

Figure 2 An example service



Service Description

Within a service-oriented architecture, applications are composed of services that interact by exchanging messages. Web Service Description Language (WSDL) documents are used to describe services. The WSDL documents specify the messages that are required to access a service. TIBCO ActiveMatrix BusinessWorks provides a WSDL palette for creating or importing WSDL files into your project. See the TIBCO ActiveMatrix BusinessWorks documentation for more information about WSDL files.

A service has the following parts:

- Endpoints
- Interface
- Partner Bindings

The following sections describe each part.

Endpoints

Endpoints expose the service to other applications and services. An endpoint is analogous to a port in a WSDL file. You can create multiple endpoints for each service so that operations within the implementation of the service can be invoked in more than one way.

TIBCO ActiveMatrix BusinessWorks BPEL Extension supports SOAP or Local endpoints. Simple Object Access Protocol (SOAP) is a standard protocol for invoking web services. Local endpoints provide a highly efficient mechanism to invoke locally-defined services within the same TIBCO ActiveMatrix BusinessWorks project.

Interface

The interface describes the operations available within the service. The interface is analogous to a portType in a WSDL file. Each interface can contain multiple operations. The implementation of each operation is an orchestration process created with the ActiveMatrix BusinessWorks BPEL Extension.

Partner Links

Partners are other services that your orchestration process can invoke. For example, the Return Merchandise Authorization service must invoke the Shipping Quotes service to obtain a quote for return shipping cost.

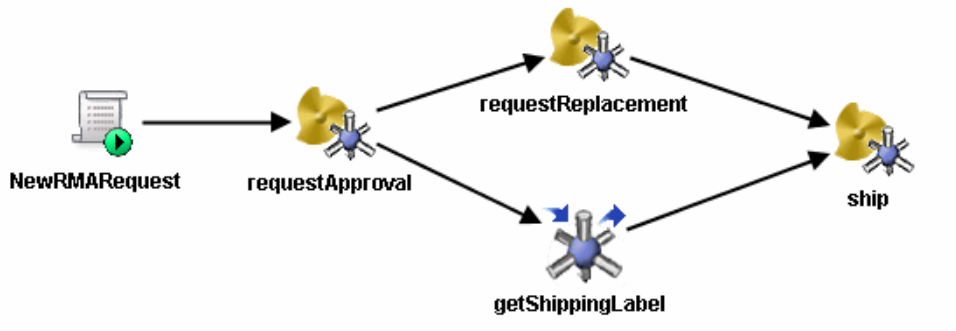
Instead of specifying the actual endpoint of a partner service when you define an orchestration process, you may wish to specify the interface, and then let the service that uses the orchestration process bind the interface to the actual endpoint. This allows you to easily change third-party partner services without changing your implementation.

Partner Link Configuration resources associate partners with endpoints. Your service can then correlate partners invoked by the operations with partner link configurations that specify the actual bindings to endpoints.

Orchestration Processes

The BPEL specification describes a standard language for defining business orchestration processes. An orchestration process is a flow of work that accomplishes some task. The orchestration process starts with an incoming message and continues with activities that perform the work of your business process. Orchestration processes provide the implementation for operations within your service-oriented architecture. For example, the Return Merchandise Authorization service has several operations. Figure 3 illustrates an example orchestration process that provides the implementation of the approveRMA operation.

Figure 3 An example orchestration process



This example illustrates the following flow of activities:

1. Receive a new RMA request.
2. Invoke a TIBCO ActiveMatrix BusinessWorks automated process to obtain the required approvals for the RMA request.
3. In parallel, perform the following tasks:
   a. Invoke a TIBCO ActiveMatrix BusinessWorks automated process to obtain replacement merchandise from inventory, if necessary.
   b. Invoke a partner service to obtain a shipping label for the returned merchandise.
4. Invoke a TIBCO ActiveMatrix BusinessWorks automated process to perform the shipment of the replacement merchandise (if necessary) and the shipping label.

Activities

Orchestration processes describe the flow of the tasks that must be accomplished. The implementation of the services and automated business processes invoked by an orchestration process are provided by other resources inside or outside of the project.

There are significantly fewer types of activities that can be placed in an orchestration process than in a TIBCO ActiveMatrix BusinessWorks automated business process. See Activities Within Orchestration Processes on page 51 for more information about the activities that can be performed in an orchestration process.

Process Variables

Similar to TIBCO ActiveMatrix BusinessWorks automated business processes, orchestration processes can define and use process variables to store data for processing. See Process Variables on page 55 for more information about process variables.

Correlations

Correlations provide a customizable mechanism for making sure that incoming messages are routed to the correct executing instance of an orchestration process. Typically correlations are used in distributed applications where services are invoked asynchronously over a significant period of time (hours or days). For example, you may use a purchase order ID to keep track of messages that should be routed to a certain process. That is, all communication (invoices, emails, and so on) about the same purchase order will contain the purchase order ID in the message. The TIBCO ActiveMatrix BusinessWorks BPEL Extension engine automatically routes messages to the correct process instance based on the correlation sets you define. See Correlations on page 62 for more information.

Groups

Groups in orchestration processes are defined in a similar manner to groups in TIBCO ActiveMatrix BusinessWorks process definitions. However, groups in an orchestration process are equivalent to structured activities in WS-BPEL. Structured activities are used to control the execution flow of the orchestration process. Groups in an orchestration process can also have their own variables, correlations, error handling routines, or event handling routines.

The following types of groups are available in the ActiveMatrix BusinessWorks BPEL Extension:

- Scope — equivalent to a <scope> in a WS-BPEL process. This type of group provides a way to encapsulate a set of activities so that the contained activities can have their own variables, correlations, error handling or event handling routines.
- Iterate — similar in cocept to a <forEach> structured activity in a WS-BPEL process. However, his type of group iterates over the values contained within a variable instead of looping through an index.
- Pick First — equivalent to a <pick> structured activity in a WS-BPEL process. This type of group waits for the occurrence of one event from a set of specified events.
- While True — equivalent to a <while> structured activity in a WS-BPEL process. This type of group executes a set of activities repeatedly until the specified condition is met.
- If — equivalent to an <if> structured activity in a WS-BPEL process. This type of group allows you to specify a series of ordered conditions and then execute the first path whose condition evaluates to true.

See Overview of Groups on page 68 for more information about groups.

Event Handling

Orchestration processes can define event handling routines that perform processing if the specified event occurs. For example, you can set an alarm in the process to go off every day so that the status of the process is logged each day. Another example of event handling is receiving a message that requests the current status of the process. Event handling is performed by the On Alarm and On Event activities within an orchestration process. See On Alarm on page 140 and On Event on page 143 for more information. Also the examples in Chapter 5, Scopes, Iteration, and Conditional Processing Using Groups, on page 67 describe how event handling routines work within groups inside of an orchestration process.

Exception Handling

Orchestration processes can define exception handling routines that perform processing when an exception is thrown. See Chapter 6, Exception Handling, on page 85 for more information.

Testing and Deployment

TIBCO ActiveMatrix BusinessWorks provides a framework for testing and deploying projects. The ActiveMatrix BusinessWorks BPEL Extension uses this framework to test and deploy orchestration processes. See Chapter 8, Testing and Deploying Orchestration Processes, on page 103 for more information.

Exporting and Importing

TIBCO ActiveMatrix BusinessWorks BPEL Extension orchestration processes are based on the WS-BPEL standard (http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v20-rddl.html). WS-BPEL defines a standard notation for expressing business processes in XML. TIBCO ActiveMatrix BusinessWorks BPEL Extension supports exporting and importing orchestration processes as standard WS-BPEL 2.0 files. WSDL and XSD files associated with orchestration processes are also exported when an orchestration process is exported.

TIBCO ActiveMatrix BusinessWorks BPEL Extension provides functionality beyond the WS-BPEL standard. For example, the WS-BPEL standard does not define how to graphically represent business processes. Also, TIBCO ActiveMatrix BusinessWorks BPEL Extension provides additional functionality such as process checkpointing and writing information to a log file. When a project is exported, these extensions to WS-BPEL functionality are handled in a way that complies with the standard. Any project previously exported from the ActiveMatrix BusinessWorks BPEL Extension can be re-imported into a ActiveMatrix BusinessWorks BPEL Extension project. However, it is not guaranteed that TIBCO extensions will be recognized by other WS-BPEL-based products.

See Chapter 9, Exporting and Importing Orchestration Processes, on page 109 for more information.

Chapter 2

Installing TIBCO ActiveMatrix BusinessWorks BPEL Extension

This chapter describes how to install TIBCO ActiveMatrix BusinessWorks BPEL Extension.

Topics

Installer Overview

The installer allows you to run in different modes. Each mode is supported on all platforms.

- GUI mode
- Console mode
- Silent mode

GUI Mode

In GUI mode, the installer presents panels that allow you to make choices about product selection, product location, and so on. When you invoke the installer by double-clicking on the icon, GUI mode is used.

Console Mode

Console mode allows you to run the installer from the command prompt or terminal window. This is useful if your machine does not have a Windows environment.

Silent Mode

Silent mode either installs using default settings or uses a response file that was saved during an earlier installation. Silent mode installs without prompting you for information.

- If no response file has been recorded earlier and you invoke the installer with the `-silent` argument, the default installation parameters are used.
- If a response file exists, and the installer is started with `-options` <*responseFileName*> as an argument, the installer uses the values specified by the user when the response file was generated.

Uninstalling TIBCO ActiveMatrix BusinessWorks BPEL Extension

If another product is dependent on the product you wish to uninstall, you are informed that you must uninstall the other product first.

Microsoft Windows

Use one of the following to uninstall TIBCO ActiveMatrix BusinessWorks BPEL Extension:

- Navigate to the `BW_HOME\bwbpel\_uninst` directory located in the TIBCO ActiveMatrix BusinessWorks home directory and invoke the `Tibuninstall.exe` program.
- Use Add/Remove Programs from the Control Panel.

UNIX Systems

Navigate to the `_uninst` directory located in the TIBCO ActiveMatrix BusinessWorks home directory and invoke the `Tibuninstall.exe` program.

Installation Registry

The installer maintains an installation registry. The registry location depends on the platform. This section explains where the registry files are located. The files have vpd as a prefix, which stands for Vital Product Database.

Do not edit, modify, rename, move, or remove any of the registry vpd files.

Microsoft Windows Platforms

TIBCO products maintain the installation registry in the *%SystemRoot%* directory. The following files represent the installation registry:

%SystemRoot%`\vpd.properties`
%SystemRoot%`\vpd.properties.tibco.`*systemName*

Installer Disk Space Requirements in Temporary Area

The entire package is extracted into a temporary folder, typically `SystemDrive:\Temp` or `SystemDrive:\Documents and Settings\<`*user_name*`>\Local Settings\Temp`.

The installer requires 40MB of free space in the temp directory.

UNIX Platforms

The installation registry is maintained in the following files in the user's home directory:

`User_Home_Directory/vpd.properties`
`User_Home_Directory/vpd.properties.tibco.`*systemName*

If installation is performed by super-user (root), the installation registry is maintained as follows:

- On Solaris and HP-UX, in the root user's home directory (which is /) as two vpd files.
- On Linux, in the /root directory as two vpd files.
- On AIX, in the /usr/lib/objrepos directory as two vpd files.

Installer Disk Space Requirements in Temporary Area

The installer launcher first extracts a Java Virtual Machine (JVM) in a temporary directory and uses this JVM to launch itself. The size of the extracted JVM differs from platform to platform.

On UNIX platforms the following disk space is required in the temporary area:

- On Solaris, 50 MB of free disk space in `/var/tmp`
- On HP-UX, 85 MB of free disk space in `/var/tmp`
- On AIX, 50 MB of free disk space in `/tmp`
- On Linux, 50 MB of free disk space in `/tmp`

If your system does not have sufficient free disk space in the above temporary area, you can still run the installer with a different temporary area by using the following option when starting the installer:

<*install_package_name*>`.bin -is:tempdir /new_tmp`

where `/new_tmp` has sufficient free disk space.

Disk Space Requirement in User's Home Directory

On UNIX platforms, when a regular (non-root) user installs a TIBCO product, the installation registry (two vpd files) is maintained in the user's home directory. As more products are installed, entries are added into these vpd files.

The user's home directory must have at least have 500 KB of free disk space.

Installation History

The installer and uninstaller creates a file called `TIBCOInstallationHistory.xml` in the same location where the installation registry is created. Each time an installation and uninstallation is performed, entries are appended to the file.

On Windows: *%SystemRoot%*`\TIBCOInstallationHistory.xml`

On UNIX: `Users_Home_Directory/TIBCOInstallationHistory.xml`

The file `TIBCOInstallationHistory.xml` therefore contains the record of all installation and uninstallation activities of all products, features and components.

Do not edit, modify, rename, move, or remove the `TIBCOInstallationHistory.xml` file.

Required and Optional TIBCO Products

Depending on the tasks you wish to perform, you must install one or more other TIBCO products. The next table describes required and optional products and their purpose.

Table 2 Required and Optional TIBCO Products

Component	Purpose
TIBCO ActiveMatrix BusinessWorks	Required. TIBCO ActiveMatrix BusinessWorks requires other software such as TIBCO Runtime Agent and TIBCO Designer. See *TIBCO ActiveMatrix BusinessWorks Installation* for more information about the required products for TIBCO ActiveMatrix BusinessWorks.
TIBCO Administrator	Required. TIBCO Administrator includes the following modules: • User Management. Management of authentication, roles and users, that is, connecting roles (groups) and users to access control lists (ACLs). This includes security for deployed applications at runtime. • Resource Management. Monitoring of machines and of all running applications in a TIBCO administration domain. Alerts can be created, for example, to notify an administrator if the number of processes or disk usage exceed a certain number. • Application Management. Uploading of Enterprise Archive (EAR) files, creation, configuration, deployment, and monitoring of applications. This console is also used to start and stop applications. TIBCO Administrator is available as a separate installation and can be installed after installing TIBCO ActiveMatrix BusinessWorks BPEL Extension.
TIBCO Enterprise Message Service	Optional. TIBCO Enterprise Message Service allows you to use the Java Message Services (JMS) as the message transport. TIBCO Enterprise Message Service is available as a separate installation and can be installed after TIBCO ActiveMatrix BusinessWorks BPEL Extension is installed.

Installing on Microsoft Windows

Before starting the installation procedure, review the topics in this section to determine that your system meets the basic requirements and that you have the prerequisite software installed.

The following is a list of prerequisites for installing TIBCO ActiveMatrix BusinessWorks BPEL Extension on Microsoft Windows systems. See Installer Disk Space Requirements in Temporary Area on page 16 for additional disk space requirements.

Table 3 Supported platforms for Microsoft Windows

Platform	Service Pack/Edition	Disk Space (MB)
Microsoft Windows 2000	Service Pack 2	30
Microsoft Windows XP	Professional Edition	30
Microsoft Windows Server 2003	Enterprise Edition for server-class hardware, Standard Edition for desktops	30

TIBCO ActiveMatrix BusinessWorks Must be Installed Before TIBCO ActiveMatrix BusinessWorks BPEL Extension

Before you can install TIBCO ActiveMatrix BusinessWorks BPEL Extension, you must install TIBCO ActiveMatrix BusinessWorks. If you select the Typical option during installation, the installer places all libraries and other products required by TIBCO ActiveMatrix BusinessWorks BPEL Extension into the `BW_HOME` directory.

During installation, the installer checks for the availability of all dependent products in the target system. If any of the dependencies are not available, the installer immediately exits. Otherwise installation proceeds.

Installer Account

You must have administrator privileges for the machine on which TIBCO ActiveMatrix BusinessWorks BPEL Extension is installed.

If you do not have administrator privileges, the installer exits. You must then log out of the system and log in as a user with the required privileges, or request your system administrator to assign the privileges to your account.

Installing from Network Drive

If you intend to install the product on a network drive, you must ensure that the account used for installation has permission to access the network drive.

Installing on Windows 2000 Terminal Server

There are two modes in terminal server, `Execute` and `Install`. By default all users are logged on in `Execute` mode, which allows them to run the applications. When you want to install TIBCO ActiveMatrix BusinessWorks BPEL Extension for use by everyone, the Administrator should change to `Install` mode.

The best way to install TIBCO ActiveMatrix BusinessWorks BPEL Extension is to use the Add/Remove Programs control panel applet, because this automatically sets the mode to `Install` during the installation and then back to `Execute` at the end. Alternatively, you can manually change your mode to `Install` by typing:

```
C:\> change user /install
```

Change back to execute:

```
C:\> change user /execute
```

Check your current mode:

```
C:\> change user /query
```

If you install in the `Execute` mode, the installation registry is maintained in your user home directory. If you install in the `Install` mode, the installation registry is maintained in the *%SystemRoot%* folder.

Installing TIBCO ActiveMatrix BusinessWorks BPEL Extension on Microsoft Windows

Two installation types are available: Development Tools and Customize Installation. The Development Tools type installs all the products in the package on the specified platform. The Customize Installation type allows you to select one or more product features to be installed.

Use one of the following modes to install TIBCO ActiveMatrix BusinessWorks BPEL Extension.

Installing in GUI mode

To install the product in GUI mode:

1. Open the physical media or download the TIBCO ActiveMatrix BusinessWorks BPEL Extension package. For example, `TIB_bwbpel_5.9.0_win_x86.zip` for Microsoft Windows platform.

2. Extract the product's archive file to a temporary directory.
3. Navigate to the temporary directory.
4. Run **TIBCOUniversalInstaller**. You can do so in one of the following ways:
   - Double-click the installer icon.
   - On the command prompt, provide the absolute path of the installer file on without specifying any options. The installer defaults to GUI mode.
5. The Welcome screen appears. Click **Next**.
6. The License Agreement screen appears. After reading through the license text, click **I accept the terms of the license agreement** and then click **Next**.
7. The Installation Profile Selection screen appears. Select **Development Tools** and then click **Next**.
8. If you selected Development Tools, proceed to step 9. If you selected **Customize Installation** checkbox, then select the components available for installation. By default, all the components are selected. Clear the checkboxes next to the components you don't want installed. Click **Next**.
9. The TIBCO Installation Home screen appears. Select the **Use an existing TIBCO_HOME** option and select an environment from the drop-down list. Click **Next**.
10. The Pre Install summary screen appears. Verify the list of features selected for install and then click **Install**. The installation location of the links depends on the TIBCO Business Studio version available in your environment.
11. The Post Install Summary screen appears. This screen summarizes the installation process. Click **Finish** to complete the installation process and close the installer window.

Install Using Console Mode

Console mode allows you to install the software in a non-windows environment. The installer will prompt you for values. Type the following at the command prompt:

```
TIB_bwbpel-simple_<version_num>_win_x86.exe -is:javaconsole -console
```

When running in console mode you can move through the installation process as described next:

```
Enter Key = Moves forward in the installer
2 = Goes back to previous screen
3 = Cancels the Wizard and exits the installation or uninstallation
4 = Redisplays the current screen
```

Install Using Silent Mode

Silent mode allows you to install the software without prompts. Type the following at the command prompt:

```
TIB_bwbpel-simple_<version_num>_win_x86.exe -silent
```

Install and Generate a Response File

You can generate a response file during installation which you can later use to invoke the installer with the selected values as default values (GUI mode) or as selected values (silent mode).

To install and generate a response file, type the following at the command prompt:

```
TIB_bwbpel-simple_<version_num>_win_x86.exe -options -record
C:\<directoryPath>\<responseFile>
```

The response file does not record selections at the component level. It does record all other selections, for example, which products you wished to install.

Install Using a Response File

You can use a previously generated response file for installation. For non-silent modes, the response file determines the defaults that are presented. For silent mode, the response file determines what will be installed.

To install using a response file, type the following at the command prompt:

```
TIB_bwbpel-simple_<version_num>_win_x86.exe -options
C:\<directoryPath>\<responseFileName>
```

Combining Options

You can combine the different available options. For example, to install in silent mode using a response file, use:

```
<install_package_name>.exe -silent -options <responseFileName>
```

To install using console mode and generate a response file, use:

```
<install_package_name>.exe -is:javaconsole -console -options -record
<responseFileName>
```

Installation on UNIX Systems

Your operating system must meet the minimum patch requirements listed in Table 4. See Installer Disk Space Requirements in Temporary Area on page 17 for additional disk space requirements.

Table 4 Supported platforms, package names, patches and disk space for UNIX systems

Platform	Notes	Disk Space (MB)
Solaris 2.7, 8, 9, 10		30
HP-UX 11.0, 11i	HP-UX 11.0 and 11i for PA-RISC HP-UX 11i v2 (HP-UX 11.23) for Intel Itanium	30
AIX 5.2, 5.3		30
Linux	Requires kernel 2.4.6 or later glibc 2.2.2 or later	30

TIBCO ActiveMatrix BusinessWorks Must be Installed Before TIBCO ActiveMatrix BusinessWorks BPEL Extension

Before you can install TIBCO ActiveMatrix BusinessWorks BPEL Extension, you must install TIBCO ActiveMatrix BusinessWorks. If you choose the Typical option during installation, the installer places all libraries and other products required by TIBCO ActiveMatrix BusinessWorks BPEL Extension into the `BW_HOME` directory.

Installer Account

TIBCO 5.x products can be installed by a regular (non-root) user and super-user (root). Different users can install the same product at different locations.

Product dependencies at install time are resolved at user level through the installation registry maintained at user's home directory. See Installation Registry on page 16 for more information.

Windows Environment

A window environment such as CDE (that is, X Windows) is required to run the installer in GUI mode. It is not required for a console installation.

Installing TIBCO ActiveMatrix BusinessWorks on UNIX

After unpacking the software and accepting the license agreement, you can choose to perform a typical install or custom install.

- A typical install has minimal prompts and installs standard components in default locations.
- A custom install prompts you to choose which pieces of the product suite to install and installs only those components.

If you are installing for the first time, you must specify the installation directory where the products in this product suite will be installed. The default installation directory depends on who performs the installation:

- For root users, the default installation directory is `/opt/tibco`.
- For non-root users, the default installation directory is `/<myhome>/tibco`, where <*myhome*> is the home directory of the user.
- If installing TIBCO ActiveMatrix BusinessWorks BPEL Extension on AIX, use a non-root user account.

Use one of the following modes to install the software. The examples assume you are installing TIBCO ActiveMatrix BusinessWorks BPEL Extension on Solaris 2.7.

Install Using GUI Mode

GUI Mode allows you input values in panels. Type the following in a terminal window:

```
% ./TIB_bwbpel-simple_<version_num>_<platform>.bin
```

Install Using Console Mode

Console mode allows you to install the software in a non-windows environment. The installer prompts you for values. Type the following in a terminal window:

```
% ./TIB_bwbpel-simple_<version_num>_<platform>.bin -is:javaconsole
-console
```

When running in console mode, you can move through the installation process as described next:

```
Enter Key = Moves forward in the installer
2 = Goes back to previous screen
3 = Cancels the Wizard and exits the installation or uninstallation
4 = Redisplays the current screen
```

Install Using Silent Mode with Default Values

Silent mode allows you to install the software without prompts using default values. Type the following in a terminal window:

```
% ./TIB_bwbpel-simple_<version_num>_<platform>.bin -silent
```

Install and Generate a Response File

You can generate a response file during installation which you can later use to invoke the installer with the selected values as default values (GUI mode) or as selected values (silent mode).

To install and generate a response file, type the following at the command prompt:

```
% ./TIB_bwbpel-simple_<version_num>_<platform>.bin -options -record
/<directoryPath>/<responseFile>
```

The response file does not record selections at the component level. It does record all other selections, for example, which products you wished to install.

Install Using a Response File

You can use a previously generated response file for installation. For non-silent modes, the response file determine the defaults that are presented. For silent mode, the response file determines what will be installed.

To install using a response file, type the following at the command prompt:

```
% ./TIB_bwbpel-simple_<version_num>_<platform>.bin -options
/<directoryPath>/<responseFileName>
```

Combining Options

You can combine the different available options. For example, to install in silent mode using a response file, use:

```
% ./TIB_bwbpel-simple_<version_num>_<platform>.bin -silent -options
<responseFileName>
```

To install using Console mode and generate a response file, use:

```
% ./TIB_bwbpel-simple_<version_num>_<platform>.bin -is:javaconsole
-console -options-record <responseFileName>
```

Post Installation

All TIBCO ActiveMatrix BusinessWorks BPEL Extension users must have read, write, and execute permissions for the following directories:

```
$TIBCO_HOME/bw/<version_num>/bin
$TIBCO_HOME/bw/<version_num>/bin/logs
$TIBCO_HOME/tra/<version_num>/logs
```

For example, if TIBCO ActiveMatrix BusinessWorks BPEL Extension has been installed in `/opt/tibco`, the user who installed TIBCO ActiveMatrix BusinessWorks Extension should execute the following commands:

```
% chmod 777 /opt/tibco/bw/<version_num>/bin/
% chmod 777 /opt/tibco/bw/<version_num>/bin/logs
% chmod 777 /opt/tibco/tra/<version_num>/logs
```

In addition, the palettes directory (`$TIBCO_HOME/bw/<version_num>/lib/palettes`) must be writable by all users who will use TIBCO Designer. Users must have write permission for this directory to be able to use the Java activity's Compile button.

Alternatively, change the Java activity's compilation directory by changing the value of `java.property.javaCode` in the `.tra` file that is used to start TIBCO ActiveMatrix BusinessWorks (`designer.tra` for the TIBCO Designer GUI or the appropriate `.tra` file for the engine if running from a deployment).

Chapter 3 TIBCO ActiveMatrix BusinessWorks BPEL Extension Tutorial

This chapter walks you through the simple example project included in TIBCO ActiveMatrix BusinessWorks BPEL Extension. The tutorial is designed to familiarize you with how the resources in this product work together to create business services.

Topics

Overview of the Tutorial

The tutorial is included in the *BW_HOME*`/bwpel/examples` directory. This directory contains two files:

- `tutorial.zip` — contains the sample project repository. Unzip this file and open the sample project in TIBCO Designer to view the tutorial project.
- `tutorial_complete.zip` — You can use this file to compare your tutorial project with the intended result.

The tutorial project is a simplified online mortgage broker application. The user requests a loan through the broker, and the broker processes the loan request using one of three third-party partner services. The tutorial is designed to walk you through the basic operations of creating a service in TIBCO ActiveMatrix BusinessWorks BPEL Extension.

The tutorial project is partially completed. The sections of this chapter describe how to configure the resources in the project to become a working project.

This chapter describes each component of the tutorial in the following sections:

WSDL Files in the Tutorial

The tutorial involves a broker service and three potential third-party services. Each service requires a WSDL file that contains the interface and schemas of the service.

Mortgage Broker WSDL

The `BrokerWSDL` file of the mortgage broker service is stored in the `tutorial/MortgageBroker` folder of the project. The following is the WSDL code for this service:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:tns="http://xmlns.example.com/BrokerWSDL"
xmlns:ns="http://www.tibco.com/schemas/Untitled/MortgageBroker/Sch
ema.xsd" targetNamespace="http://xmlns.example.com/BrokerWSDL">
   <import namespace=
      "http://www.tibco.com/schemas/Untitled/MortgageBroker/
       Schema.xsd" location="../Metadata/LoanSchema.xsd"/>
   <message name="MessageIn">
      <part name="loanRequest" element="ns:LoanRequest"/>
   </message>
   <message name="MessageOut">
      <part name="loanReply" element="ns:LoanReply"/>
   </message>
   <message name="MessageFault">
      <part name="fault" element="ns:Error"/>
   </message>
   <portType name="BrokerPT">
      <operation name="requestLoanOperation">
         <input message="tns:MessageIn"/>
         <output message="tns:MessageOut"/>
         <fault name="fault1" message="tns:MessageFault"/>
      </operation>
   </portType>
</definitions>
```

The mortgage broker WSDL contains one port type, `BrokerPT`. The `BrokerPT` port type contains one operation, `requestLoanOperation`. This operation has an input, output, and fault message.

Partner WSDLs

The mortgage broker service will call one of three partner third-party services from Bank_A, Bank_B, or Bank_C. Each of these third-party services implements the same abstract interface. Because the third-party services have agreed to implement the same interface, the mortgage broker service can switch between the third-party services without changing the implementation of the mortgage broker service.

The abstract interface used by the third-party services is contained in the `tutorial/Metadata` folder of the project. The `Metadata` folder contains the `BankWSDLAbstract` WSDL file and the `LoanSchema` XSD referenced by the WSDL. The following is the `BankWSDLAbstract` WSDL file:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ns="http://www.tibco.com/schemas/Untitled/MortgageBroker/Sch
ema.xsd" xmlns:tns="http://xmlns.example.com/BankWSDL"
targetNamespace="http://xmlns.example.com/BankWSDL">
   <documentation>Common service interface to request for a
      loan from a bank</documentation>
   <import namespace=
      "http://www.tibco.com/schemas/Untitled/MortgageBroker/
      Schema.xsd" location="LoanSchema.xsd"/>
   <message name="MessageIn">
      <part name="loanRequest" element="ns:BrokerLoanRequest"/>
   </message>
   <message name="MessageOut">
      <part name="loanReply" element="ns:LoanReply"/>
   </message>
   <message name="MessageFault">
      <part name="fault" element="ns:Error"/>
   </message>
   <portType name="BankPT">
      <operation name="loanApprovalOperation">
         <input message="tns:MessageIn"/>
         <output message="tns:MessageOut"/>
         <fault name="fault1" message="tns:MessageFault"/>
      </operation>
   </portType>
</definitions>
```

The `BankWSDLAbstract` contains one port type, `BankPT`. The `BankPT` port type contains one operation, `loanApprovalOperation`. This operation has an input, output, and fault message. See the `LoanSchema.xsd` file in the `Metadata` folder of the project for a description of the structure of the `BrokerLoanRequest`, `LoanReply`, and `Error` messages.

Each third-party service publishes a concrete WSDL file with the bindings to the endpoint where the service can be invoked. Each concrete WSDL file references the `BankWSDLAbstract` WSDL. The mortgage broker service stores the third-party partner service concrete WSDL files in the `tutorial/MortgageBroker/PartnerWSDLs` folder of the project. For example, the following is the concrete WSDL for the Bank_A service:

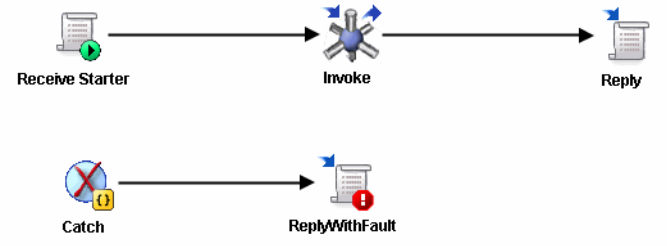
```
<?xml version = "1.0" encoding = "UTF-8"?>

<!--Created by TIBCO WSDL-->
<wsdl:definitions targetNamespace =
"http://xmlns.example.com/Bank_A" xmlns:ns0 =
"http://xmlns.example.com/BankWSDL" xmlns:soap =
"http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns =
"http://xmlns.example.com/Bank_A" xmlns:wsdl =
"http://schemas.xmlsoap.org/wsdl/">
   <wsdl:import location = "../../Metadata/BankWSDLAbstract.wsdl"
namespace = "http://xmlns.example.com/BankWSDL"/>
   <wsdl:types/>
   <wsdl:service name = "Bank_A_Service">
      <wsdl:port binding = "tns:BankPTEndpoint2Binding" name =
"BankPTEndpoint2">
         <soap:address location =
"http://localhost:9091/ExternalServices/Bank_A/Bank_A_Service.serv
iceagent/BankPTEndpoint2"/>
      </wsdl:port>
   </wsdl:service>
   <wsdl:binding name = "BankPTEndpoint2Binding" type =
"ns0:BankPT">
      <soap:binding style = "document" transport =
"http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name = "loanApprovalOperation">
         <soap:operation soapAction =
"/ExternalServices/Bank_A/Bank_A_Service.serviceagent/BankPTEndpoi
nt2/loanApprovalOperation" style = "document"/>
         <wsdl:input>
            <soap:body parts = "loanRequest" use = "literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body parts = "loanReply" use = "literal"/>
         </wsdl:output>
         <wsdl:fault name = "fault1">
            <soap:fault name = "fault1" use = "literal"/>
         </wsdl:fault>
      </wsdl:operation>
   </wsdl:binding>
</wsdl:definitions>
```

Configuring the Orchestration Process

The mortgage broker orchestration process stored in the `tutorial/MortgageBroker` folder of the project is named `LoanRequestProcess`. Figure 4 illustrates this process.

Figure 4 Mortgage broker orchestration process



The process performs the following actions:

1. Receives a request for a loan from a customer.
2. Invokes one of the three partner services to request a loan from a loan provider.
3. Sends a reply to the customer with the result of the loan request.
4. If an error is encountered during processing, the Catch activity is executed and a reply is sent to the customer detailing the fault that was encountered.

The following sections describe the configuration of the orchestration process and the activities within the process.

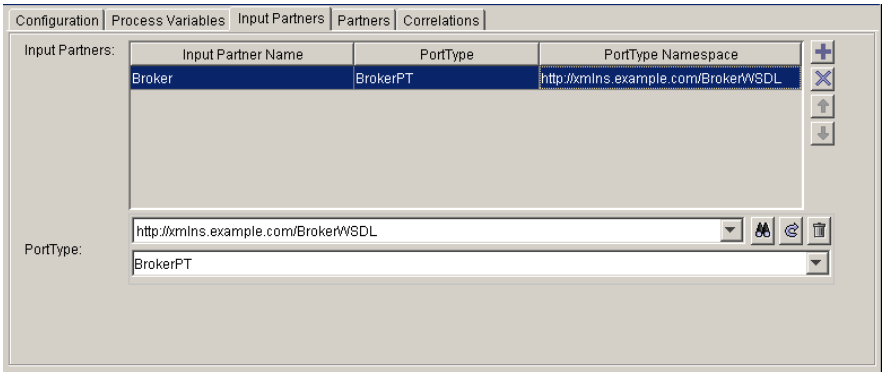
Configuring the Orchestration Process

Click the `LoanRequestProcess` resource in the project tree to show the configuration panel of the process. The two tabs that must be configured for this tutorial are the Input Partners tab and the Partners tab.

Input Partners Tab

The Input Partners tab defines the port type that this orchestration process implements. In this example, the port type is the `BrokerPT` in the `BrokerWSDL` file. Figure 5 illustrates the fields of the Input Partners tab.

Figure 5 The Input Partners tab for the mortgage broker tutorial



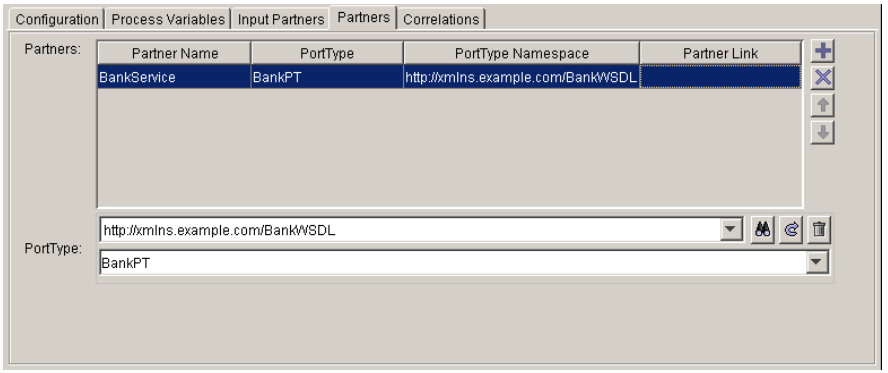
To configure the input partners for this tutorial:

1. Click the **+** button on the right side of the Input Partners field.
2. Double click on the value in the Input Partner Name field and specify `Broker` as the name for the input partner.
3. Click the **Browse** button on the right side of the Port Type field to select the WSDL file that contains the port type of this input partner. In this example, the WSDL file is `tutorial/MortgageBroker/BrokerWSDL`.
4. Select the correct port type in the WSDL file. In this example, there is only one port type, `BrokerPT`, so it is automatically selected.
5. Click **OK** to accept the correct port type and dismiss the Select a Resource dialog.
6. Click **Apply** to apply the changes to the Input Partners tab.

Partners Tab

The Partners tab defines the partner services that the orchestration process can invoke. In this example, the process can invoke any of the third-party services that implement the abstract `BankPT` port type. Figure 6 illustrates fields of the Partners tab.

Figure 6 The Partners tab for the mortgage broker tutorial



To configure the partners for this tutorial:

1. Click the **+** button on the right side of the Partners field.
2. Double click on the value in the Partner Name field and specify `BankService` as the name for the partner.
3. Click the **Browse** button on the right side of the Port Type field to select the WSDL file that contains the port type of this partner. In this example, the WSDL file is `tutorial/Metadata/BankWSDLAbstract`.
4. Select the correct port type in the WSDL file. In this example, there is only one port type, `BankPT`, so it is automatically selected.
5. Click **OK** to accept the correct port type and dismiss the Select a Resource dialog.
6. Click **Apply** to apply the changes to the Partners tab.

In this example, the orchestration process references the port type of any of the partners that can be invoked. Later in the tutorial, the partner will be bound to a specific concrete implementation using a Partner Link Configuration resource. Using this type of resource allows you to switch partners without changing your orchestration process implementation.

Configuring the Activities

Each activity in the orchestration process must be configured to perform the desired action. Click each activity in the design panel to display the configuration panel for the activity. This section describes the configuration of each activity.

Receive Starter

The Receive Starter activity receives the loan request message from the customer. Perform the following to configure the activity:

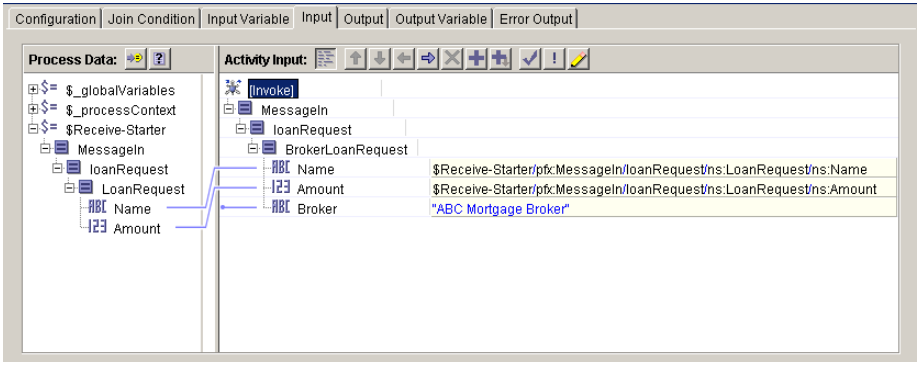
1. Select `Broker` in the Input Partner field, if it is not automatically selected.
2. Select `requestLoanOperation` in the Operation field.
3. Click **Apply** to accept the changes to the configuration.

Invoke

The Invoke activity invokes the third-party partner service. Perform the following to configure the activity:

1. Select `BankService` in the Partner field, if it is not automatically selected.
2. Select `loanApprovalOperation` in the Operation field.
3. Click `Apply` to accept the changes to the configuration.
4. Typically BPEL activities use a variable to hold data to input into the activity. TIBCO ActiveMatrix BusinessWorks BPEL Extension can also use the input mapping functionality of TIBCO ActiveMatrix BusinessWorks. In this tutorial, the input mapping mechanism is used in the Invoke activity. Click the Input tab create the input mappings illustrated in Figure 7.

Figure 7 Invoke activity input configuration



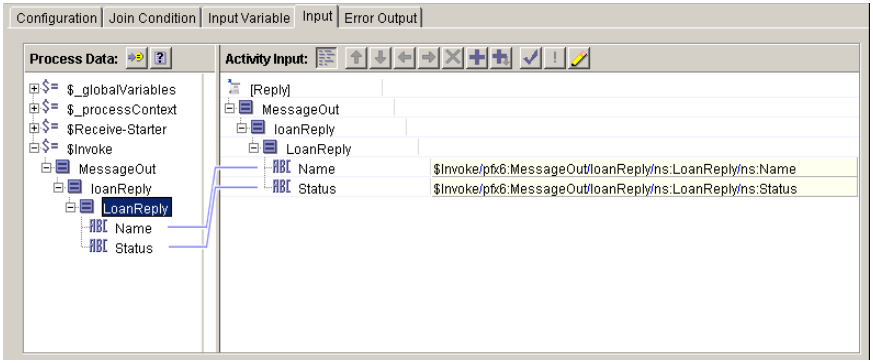
5. Click **Apply** to accept the changes to the configuration.

Reply

The Reply activity sends a reply to the user whether the loan was accepted or rejected. The TIBCO ActiveMatrix BusinessWorks BPEL Extension provides a Quick Configuration option for reply messages. This tutorial uses this option instead of the Full Configuration option. Perform the following to configure the activity:

1. Select the Receive Starter activity in the ReplyEvent field, if it is not already selected. This specifies that the reply should be directed to the sender of the message that was received by the Receive Starter activity.
2. The Input tab of the Reply activity is used to map the contents of the reply from the results of the Invoke activity. Click the Input tab to create the input mappings illustrated in Figure 8.

Figure 8 Reply activity input configuration

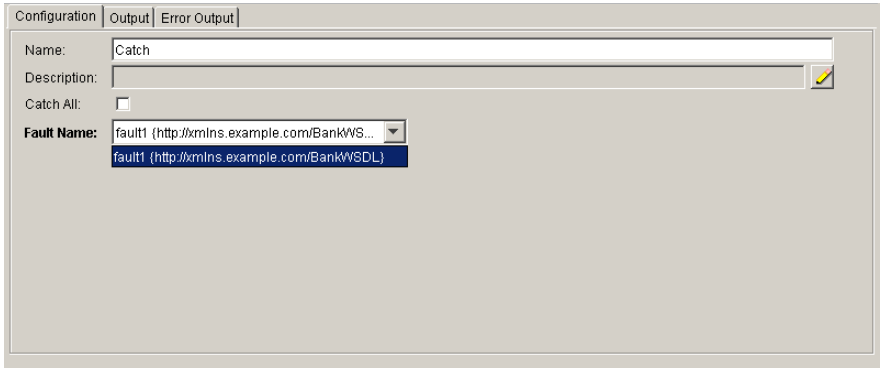


3. Click **Apply** to accept the changes to the configuration.

Catch

The Catch activity starts an error-handling routine that executes in the event that an error is encountered during processing. Catch activities can be configured to catch a specific error or any unhandled error. In this example, the Invoke activity can receive a fault message from the third-party bank service. Select fault1 in the Fault Name field as illustrated in Figure 9, if it is not already selected. This is the fault that the Invoke activity can throw (see the Error Output tab of the Invoke activity to view the schema of fault1).

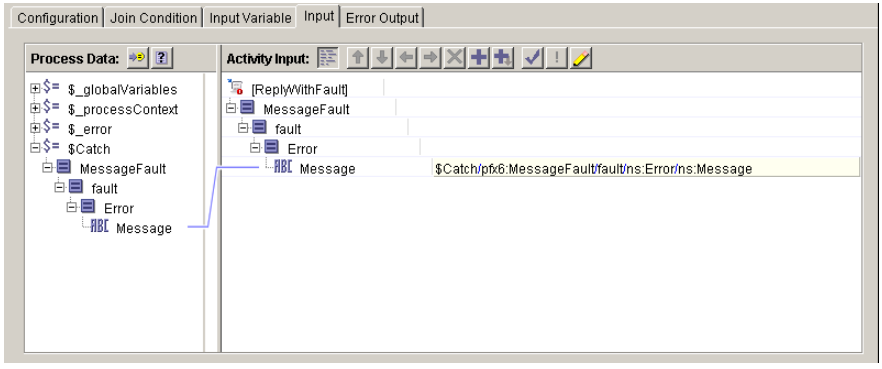
Figure 9 Catch activity Configuration tab



Reply With Fault

The Reply With Fault activity replies to a message with a fault indicating an error was encountered. In this example, the Configuration tab of the Reply With Fault activity is set to reply to the Receive Starter activity (similar to the Reply activity above). Also, the input of the Reply With Fault activity is set to return the fault captured by the Catch activity. Create input mappings as illustrated in Figure 10.

Figure 10 Reply With Fault input configuration



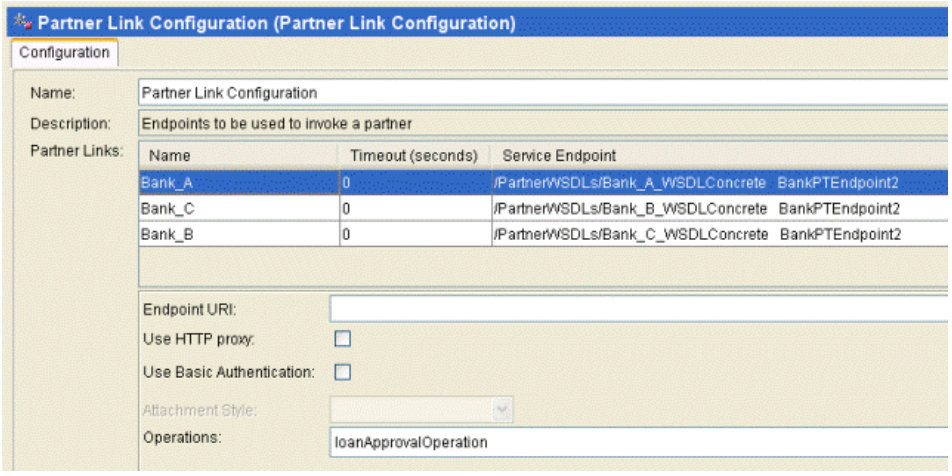
Configuring the Service

The mortgage broker service is defined by the `tutorial/MortgageBroker/BrokerService` resource. This resource provides the configuration for the mortgage broker interface, the endpoint that exposes the service to the public, the orchestration process that provides the implementation for the service's operation, and the bindings to the specific partner services used by the mortgage broker service.

Creating the Partner Link Configuration

The `tutorial/MortgageBroker/Partner Link Configuration` resource specifies the relationship between named partners and actual partner endpoints. This resource allows you to change partner endpoints or select different partners without changing your orchestration process implementation. Figure 11 illustrates the Partner Link Configuration resource for this tutorial.

Figure 11 Tutorial Partner Link Configuration resource



In this tutorial, there are three potential third-party partner services, Bank_A, Bank_B, and Bank_C. The partner services are defined in the Partner Links field. The Partner Links have already been created, but you must specify the service endpoint for each partner link to complete the configuration. Perform the following procedure to complete the configuration:

1. Double click the Service Endpoint field of the Partner Link named Bank_A.
2. Click the **Browse** button on the right side of the field to obtain a list of potential service endpoints.

3. The Select a Service Endpoint dialog appears so that you can select the WSDL file that contains the concrete service bindings of this partner. For example, for the Bank_A partner, the endpoint is `tutorial/MortgageBroker/PartnerWSDLs/Bank_A_WSDLConcrete/BankPTEndpoint2`.
4. Click **OK** to dismiss the Select a Service Endpoint dialog once you have selected the endpoint.
5. Repeat step 1 through step 4 for the Bank_B and Bank_C partner link rows.
6. Click **Apply** to apply the changes to the Configuration tab.

The `LoanRequestProcess` defined a `BankService` as the requested service. The Service resource allows you to specify which partner in the Partner Link Configuration will fill the role of the `BankService` in the orchestration process.

Endpoint Connection Resource

The mortgage broker service is exposed to the public through an HTTP endpoint that requires an HTTP Connection resource. The `tutorial/MortgageBroker/BrokerServiceHTTPConnection` defines the host and port on which this service is offered. In this example, the host is the host machine where the project executes, and the port is 9090.

This resource will be later referenced by the Endpoint Bindings tab in the Service resource described in Configuring the Service Resource on page 39.

Configuring the Service Resource

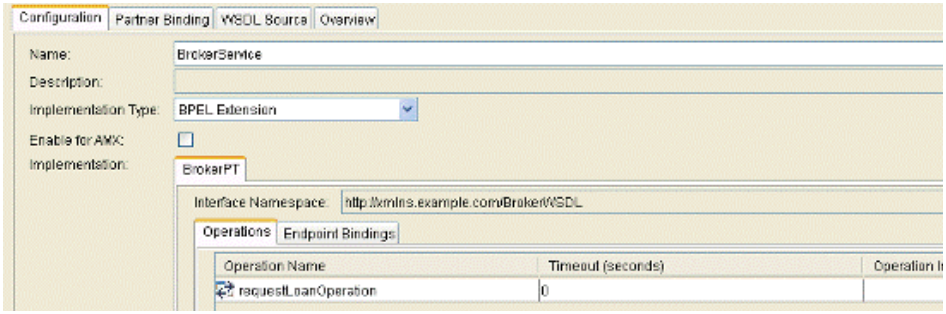
The `tutorial/MortgageBroker/BrokerService` resource is the service a user invokes to request a loan. The Service has one interface, defined in the `tutorial/MortgageBroker/BrokerWSDL` file (see Mortgage Broker WSDL on page 29). The service has already been created an partially configured. This section describes the remaining configuration steps.

Specifying the Operation Implementation

The `BrokerService` has one operation, `requestLoanOperation`. We created the orchestration process that implements this operation in Configuring the Orchestration Process on page 32. The service resource associates operations with processes that implement the operations on the Operations tab. All operations are listed in a table, and you use the Browse button in the Operation Implementation column to select the orchestration process that implements the operation. In this example, the `LoanRequestProcess` implements the `requestLoanOperation`.

Select `tutorial/MortgageBroker/LoanRequestProcess/Broker` as the implementation of `requestLoanOperation` as illustrated in Figure 12.

Figure 12 Specifying an operation implementation



Specifying the Endpoint Bindings

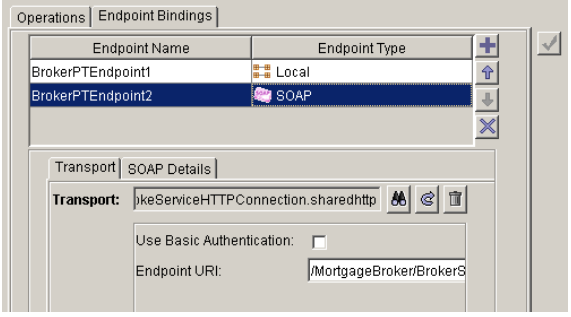
A service exposes its operations to users and other services through endpoints. Endpoints are concrete bindings to physical locations, such as an HTTP port on a web server. In this example, two endpoints have already been created for the service.

The first endpoint is local. Local endpoints are available only to resources within the same project. We will use a TIBCO ActiveMatrix BusinessWorks process definition to invoke this service by way of the local endpoint.

The second endpoint is available by way of the SOAP protocol. This endpoint would be used by client applications or other services to invoke the service over the network. For this type of endpoint, we must also specify the transport on the Transport sub-tab (described in Endpoint Connection Resource on page 39), and any other SOAP options on the SOAP Details tab.

Figure 13 illustrates the Endpoint Bindings tab.

Figure 13 Specifying endpoint bindings



Specifying the Partner Bindings

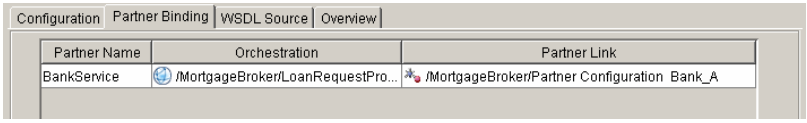
The Partner Binding tab of the service is automatically populated by any partners defined in the orchestration processes used by the service. In this example, there is one partner named `BankService`. The Partner Binding tab allows you to bind the named partner to a specific partner service in a Partner Link Configuration.

In this example, you bind `BankService` partner to the Bank_A service by way of the Partner Link Configuration defined in Creating the Partner Link Configuration on page 38.

The partner binding redirects the generic `BankService` partner to the specific partner when the service is running. If you wish to switch the partner service, you can simply change the value in the Partner Link column and redeploy the service. You do not have to change the underlying implementation to use different partner services.

Double click on the Partner Link field and click the Browse button to select `tutorial/MortgageBroker/Partner Link Configuration/Bank_A` as illustrated in Figure 14.

Figure 14 Specifying partner bindings



The External Partner Services

The `tutorial/ExternalServices` folder in the project contains the implementation of the external third-party services used in this tutorial. Normally, external services are implemented by some third-party and invoked over the internet by way of the SOAP protocol. However, because this tutorial is self-contained, the external services are implemented and stored in the tutorial project.

For simplicity, all of the external services receive a request and each one replies with a different outcome. Table 5 describes the reply message of each of the external services.

Table 5 Reply messages of each external Service

Service	Reply
Bank_A	Replies with the loan status "Rejected".
Bank_B	Replies with fault indicating "System down for maintenance".
Bank_C	Replies with the loan status "Approved".

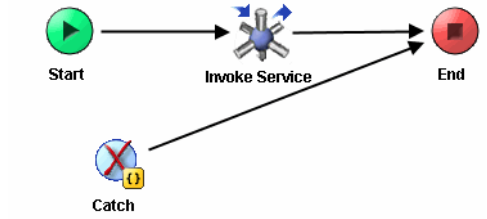
Changing the Partner Bindings on page 46 describes how to change the partner bindings of the tutorial and run the tutorial with each partner service.

Using a TIBCO ActiveMatrix BusinessWorks Process to Invoke the Service

TIBCO ActiveMatrix BusinessWorks processes can invoke ActiveMatrix BusinessWorks BPEL Extension services by way of the Invoke Service activity. To execute the mortgage broker service, you need an application to invoke the service. To accomplish this goal, the `tutorial/Borrower/LoanRequestProcess` is part of the tutorial project.

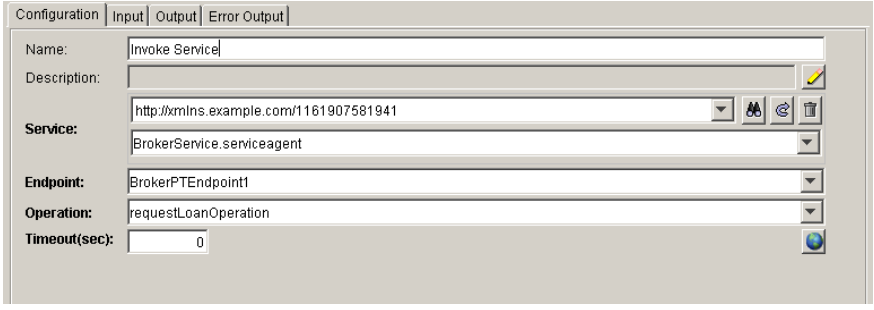
Figure 15 illustrates the `LoanRequestProcess`.

Figure 15 TIBCO ActiveMatrix BusinessWorks process definition invoking a service



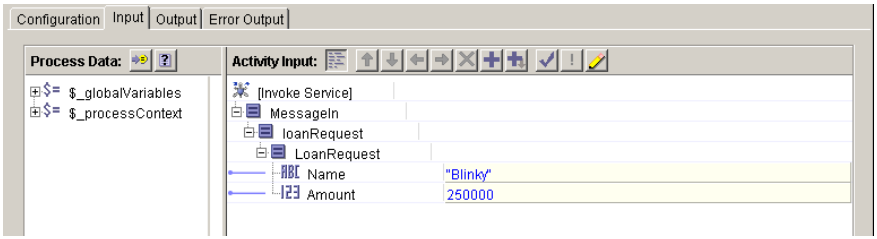
The Invoke Service activity invokes the `requestLoanOperation` over the local endpoint of the `BrokerService`. Figure 16 illustrates the Configuration tab of the Invoke Service activity. The `BrokerService` resource is selected in the Service field, and `BrokerEndpoint1` (the locally exposed endpoint) is automatically selected in the Endpoint field.

Figure 16 Configuring the Invoke Service activity



The input to the Invoke Service activity is the input message to `requestLoanOperation`. Figure 17 illustrates the input to the Invoke Service activity.

Figure 17 Configuring the Invoke Service input



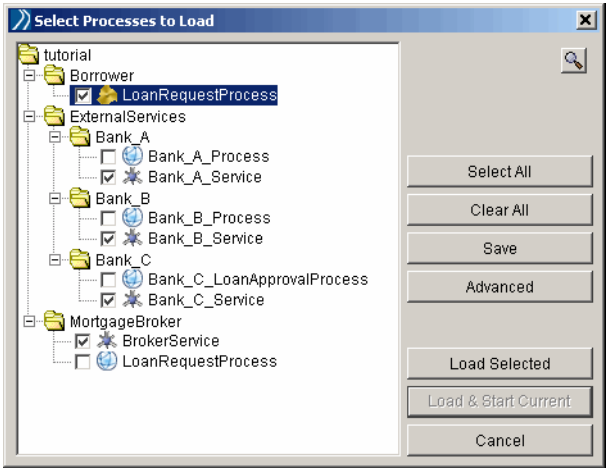
Running the Tutorial

To run the tutorial, you must start the Test mode of TIBCO ActiveMatrix BusinessWorks BPEL Extension. Click on the Tester tab of the project to enter Test mode. To run the tutorial, you must load the following resources:

- LoanRequestProcess (TIBCO ActiveMatrix BusinessWorks process that invokes the service)
- Bank_A_Service
- Bank_B_Service
- Bank_C_Service
- BrokerService

Figure 18 illustrates the Select Processes to Load dialog with the correctly selected resources.

Figure 18 Loading the resources to run the tutorial



Only one of the external services (Bank_A_Service, Bank_B_Service, or Bank_C_Service) will be used at any given time, but you can load all of the services in test mode so that you do not have to remember which one has been bound to the `BankService` partner.

Initially, we configured the mortgage broker service to invoke the Bank_A_Service. Changing the Partner Bindings on page 46 describes how to change the partner bindings to run each of the other partner services.

Changing the Partner Bindings

Once you have successfully run the tutorial project, you should attempt to change the partner bindings to a different external service and run the project again. This allows you to see different results returned by each partner service.

To change the partner bindings:

1. Click the `BrokerService` resource in the project tree.
2. Click the Partner Binding tab.
3. Double click the value in the Partner Link field then click the **Browse** button on the right side of the field.
4. Select a different partner link. For example, if you ran the project with the Bank_A partner link, choose the Bank_B partner link.
5. Click **OK** to dismiss the Select a Partner Link Dialog.
6. Click **Apply** to accept the changes to the `BrokerService` resource.
7. Save the project, then run the project in test mode again.
8. Repeat step 1 through step 7 until you have run the project using all three external services in the Partner Link field.

Chapter 4 Working with Orchestration Processes

Orchestration processes provide the implementation of service operations. This chapter describes orchestration processes.

Topics

Overview of Orchestration Processes

Orchestration processes provide the implementation of service operations to perform a given business process. An orchestration process is a flow of work that accomplishes some task. Orchestration processes are executed when an incoming message is received. For example, consider the business process illustrated in Figure 19.

Figure 19 An order fulfillment orchestration process

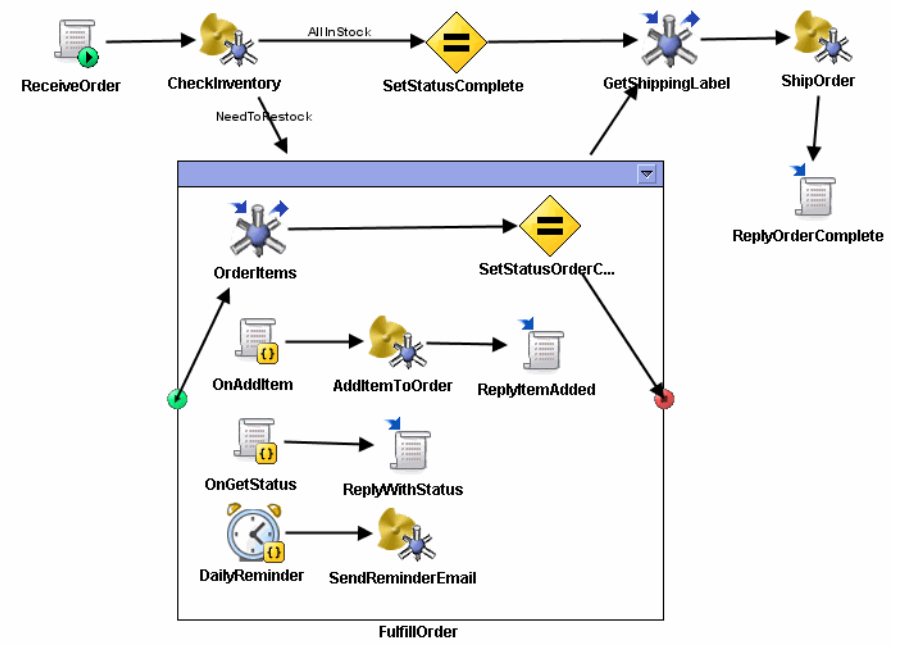


Figure 19 illustrates an order fulfillment system. An incoming order is received and inventory is checked to determine if the order can be filled. If every item ordered is in stock, the status of the order is set to complete and the process invokes a third-party service to print a shipping label. The order is then sent to shipping and a reply is sent to the requestor of the order. If one or more items is not in inventory, then a request is made to a third-party vendor to restock the items. While the request is being processed, the requestor can check the status of the order or order additional items. Also, a reminder is sent each day the order remains open to the purchasing department that the items in the order are not yet available.

Execution Path in Orchestration Processes

Orchestration processes have slightly different design and runtime semantics than TIBCO ActiveMatrix BusinessWorks process definitions. Process definitions start with a process starter and end with an End activity. Also, process definitions can have only one process starter or Start activity. Transitions are drawn from the Start activity or the process starter to the next activity in the process. Activities that are not placed within the execution path of a process definition will not execute when the process definition executes.

Orchestration processes, on the other hand, can have one or more Receive Starter activities as process starters. Also, all activities within an orchestration process have the opportunity to execute. If you place an activity in an orchestration process with no transitions to or from the activity, that activity executes when the process starts.

Typically, the Receive Starter activity is the first to execute in an orchestration process. However, some structured activities, such as a Pick First group can be executed first.

TIBCO ActiveMatrix BusinessWorks BPEL Extension does not prevent you from placing other activities before the Receive starter activity. For example, you may want to start with a Write to Log activity and then transition to the Receive Starter activity. This is permissible, but it is an extension of the WS-BPEL specification. Some activities, such as Reply or Reply With Fault should never be placed before the Receive Starter activity Therefore, you should be careful when placing activities before the Receive Starter in an orchestration process.

Orchestration processes have no End activity. Execution proceeds along the specified path until all activities along the path have been given the opportunity to execute. Once all activities within an orchestration process have had the opportunity to execute, the orchestration process ends. The Exit activity can be used to prematurely end the execution of an orchestration process, but it is not typically used as the final activity to execute in a process, like the End activity in a process definition.

Also, orchestration processes have event handling routines and error-handling routines that cannot be transitioned to or from in the main processing flow or within a scope. Execution transfers to an error-handling routine when an error occurs. Event-handling routines execute when the specified event occurs, such as an incoming message or an alarm set to go off at a particular time or for repeating intervals.

An orchestration process can only contain activities from the BPEL Extension palette. No resources from other palettes can be used within an orchestration process. Also, the BPEL Extension palette contains one activity, Invoke Service, that can be used within TIBCO ActiveMatrix BusinessWorks process definitions (not within orchestration processes) to invoke a service.

The remaining sections in this chapter as well as Chapter 5, Scopes, Iteration, and Conditional Processing Using Groups, on page 67 and Chapter 6, Exception Handling, on page 85 provide more details about the execution of orchestration processes.

Activities Within Orchestration Processes

The activities in the ActiveMatrix BusinessWorks BPEL Extension palette are the building blocks of orchestration processes. Table 6 describes each activity and its use within an orchestration process.

Table 6 Activities within orchestration processes

Activity	Description
	Assigns values to variables using XPath. Variables are useful for maintaining the status of an orchestration process or storing information to input into an activity.
	Starts an exception handling block to catch faults. Any kinds of errors encountered in an orchestration process can be caught and handled. You may want to design exception-handling routines to automatically deal with commonly encountered errors, or you may wish to use exception-handling routines to log or return detailed information.
	Saves the state of the orchestration process so that it can be restarted from the checkpoint in the event of engine failure.
	Terminates the orchestration process without executing exception handling routines.
	Invokes a TIBCO ActiveMatrix BusinessWorks process definition. TIBCO ActiveMatrix BusinessWorks allows you to create automated business processes that execute your business logic. You can invoke a TIBCO ActiveMatrix BusinessWorks process from within an orchestration process.
	Invokes a service.

Table 6 Activities within orchestration processes

Activity	Description
 Null	Performs no action. This activity can be used to join multiple execution paths.
 On Alarm	Used to perform actions based on the time within an existing orchestration process. For example, you may wish to send daily reminders that a process is running, or you may wish to exit the process after it has been running for more than 5 days.
 On Event	Used to perform actions based on an incoming message within an existing orchestration process. For example, once a new orchestration process is created to handle an incoming order, the On Event activity can wait for messages that request the status of the order.
 Receive Starter	Receives a message and starts a new job to execute the orchestration process. Typically an orchestration process starts with one or more Receive Starter activities.
 Receive	Receives a message. The orchestration process waits for the incoming message before proceeding to the next activity in the flow.
 ReplyWithFault	Replies to a message by sending a fault.
 Reply	Replies to a message.
 Rethrow	Rethrows an exception.

Table 6 Activities within orchestration processes

Activity	Description
 Sleep	Waits for the specified amount of time before continuing to the next activity in the flow.
 Throw	Throws an exception.
 Write To Log	Writes a message to a log file.

Input and Output of Activities

TIBCO ActiveMatrix BusinessWorks provides robust mapping and transformation of data for supplying input to activities within a process. Also, TIBCO ActiveMatrix BusinessWorks automatically creates process variables containing each activity's output so that the output is available to subsequently executed activities in the process. WS-BPEL orchestration processes typically use process variables for input and output of activities, and the Assign activity sets the value of the variables. TIBCO ActiveMatrix BusinessWorks BPEL Extension provides the flexibility to use either mechanism for providing input or storing output of activities within an orchestration process.

Many activities that accept input have an Input Variable tab where you can specify whether to supply a process variable containing the input or to use the Input tab for performing mapping and transformation. Many activities that supply output have an Output Variable tab where you specify the process variable in which to store the activity's output. Some activities only provide one mechanism for providing input or output. See Chapter 10, Palette Reference, on page 115 for more information about the configuration options for each activity.

See Process Variables on page 55 for more information about process variables, activity output, and using the Assign activity. See *TIBCO ActiveMatrix BusinessWorks Process Design Guide* for more information about using TIBCO ActiveMatrix BusinessWorks mapping and transformation functionality.

Process Variables

Process variables are data structures available to the activities in the orchestration process. Process variables can be used to hold data for incoming or outgoing messages as well as maintain the state of the orchestration process as it is executed. Process variables are defined and behave in the same way as TIBCO ActiveMatrix BusinessWorks process variables.

There are four types of process variables:

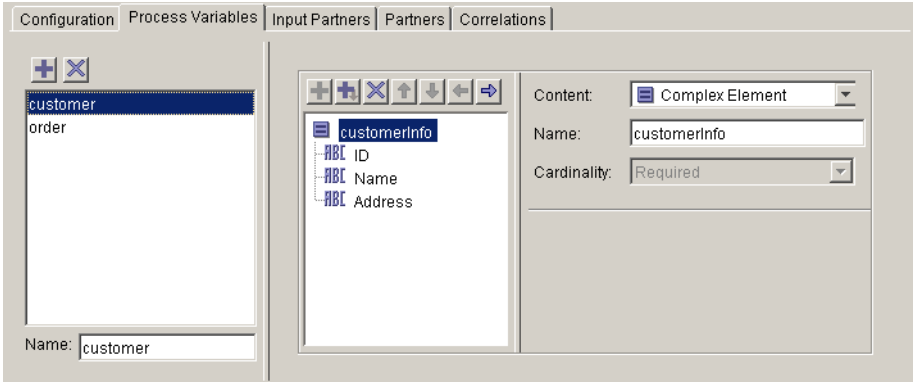
- User-Defined Process Variables
- Activity Output
- Predefined Process Variables
- Error Process Variables

User-Defined Process Variables

You can define your own process variables and assign values to them in your process definition. Process variables are defined on the Process Variables tab of the Orchestration Process resource or within a group. You create a process variable in the same way you create data schemas. See *TIBCO ActiveMatrix BusinessWorks Process Design Guide* for more information about creating data schemas.

Figure 20 illustrates creating a process variable. You add a process variable and give it a name in the left-most panel, then you create its schema in the middle panel.

Figure 20 Creating a process variable



Scope of User-Defined Process Variables

Process variables are available to activities within the scope in which the variables are defined. Process variables can be defined at the top-level orchestration process, and these variables are available to all activities within the process. Process variables can also be defined within the scope of a particular group. Process variables defined within a group are only available to activities and nested groups within the group where the variables are defined.

See Chapter 5, Scopes, Iteration, and Conditional Processing Using Groups, on page 67 for more information about groups.

Assigning a Value to a Variable

You can assign a value to a user-defined process variable either by using the Assign activity or by configuring an activity to place its output into the variable. Input and Output of Activities on page 54 discusses setting a variable using activity output configuration. This section describes the Assign activity.

The Assign activity can copy data from one process variable to another and the activity can be used to modify the value of the data before assigning it to a variable. The Assign activity can only copy or modify data, it cannot change the structure of the data. The Assign activity can include multiple assign statements, and each statement can assign values to different variables.

To use the Assign activity, you create an assign statement, then specify a source of data in the Assign From field. The Assign From field allows you to specify an XPath expression to select the data from one or more existing process variables. You can also use functions in the expression to modify the data as needed or you can obtain data from a function, such as the current time of day.

After specifying the Assign From field, you specify the target location to store the data in the Assign To field. For process variables of simple types, you can assign a value to the process variable and instantiate it in one statement. For more complex variables, you can use multiple assign statements to assign values to each field within an already instantiated process variable.

The Assign activity cannot change the structure of a variable. Therefore, you cannot do the following with the Assign activity:

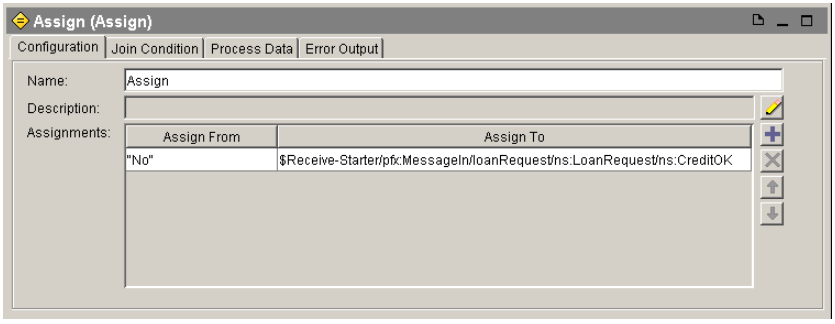
- add another element to a repeating data structure
- assign values to some but not all elements of a variable that has not yet been created
- assign a value to an element that does not exist in the variable
- any other operation that involves changing the structure of the data to which you are assigning a value

The Assign From expression must evaluate to the same datatype as the expression in the Assign To field. You cannot create a new variable by only providing some of the fields of a complex datatype in the Assign From field. You must first create a new instance of the variable by copying from a compatible data structure, then you can reassign values to particular fields in subsequent assign statements.

TIBCO ActiveMatrix BusinessWorks process definitions can perform complex mapping and construction of variables. You can use an Invoke Process activity to call a TIBCO ActiveMatrix BusinessWorks process to perform the mapping and transformation and store the output of the process in a process variable. You can use this technique to avoid the limitations of the Assign activity.

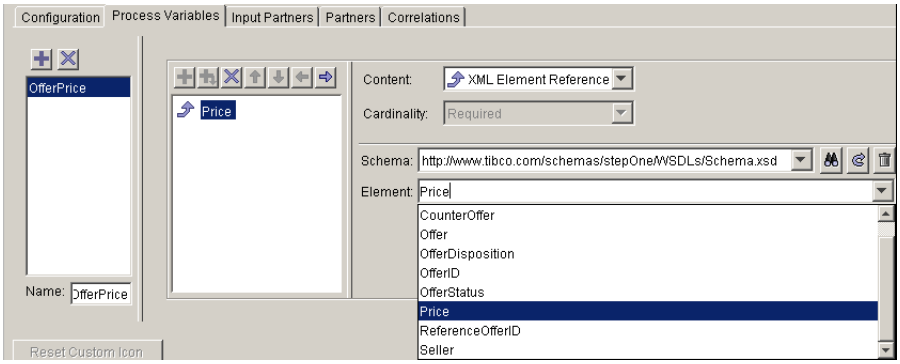
Figure 21 illustrates updating the value of a field within an existing process variable with a constant value. The `$Receive-Starter` process variable is used to hold the output of the Receive activity that accepts a new loan request. The loan request contains a requestor ID, the loan amount, and a field named CreditOK to hold status of the credit check. The illustrated Assign activity is assigning the value "No" to the CreditOK field.

Figure 21 Assigning a constant to a field of an existing process variable



The Assign activity can instantiate variables that are specified as XML Element References to simple datatypes, such as decimal, integer, or string. This is useful for storing simple values in process variables such as counters, constants, or running calculations. For example, Figure 22 illustrates a process variable named `$OfferPrice` with one element that is an XML Element Reference to an XSD element named `Price`.

Figure 22 Process variable defined from an XSD element

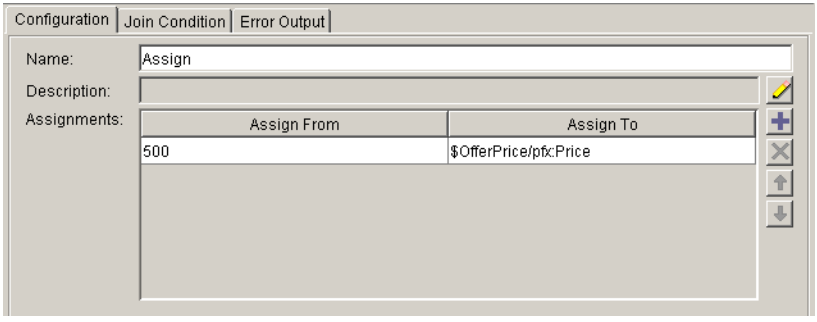


Price is defined in the XSD as:

```
<xs:element name = "Price" type = "xs:decimal"/>
```

The Assign activity can instantiate this variable in one statement by assigning a value to the variable. Figure 23 illustrates instantiating `$OfferPrice/Price` by assigning the constant "500" to the variable.

Figure 23 Instantiating a process variable of a simple datatype



See Assign on page 117 for more information about how to set the fields in the Assign activity.

Activity Output

Some activities produce output. Activities have access to any data that is output from previously executed activities in the process definition. An activity's output is placed into a process variable with the same name as the activity. You can also optionally specify a user-defined variable to hold the contents of an activity's output.

Predefined Process Variables

There are two process variables that are available to all activities that accept input: `$_globalVariables` and `$_processContext`.

`$_globalVariables` contains the list of global variables defined on the Global Variables tab of the project. See *TIBCO ActiveMatrix BusinessWorks Process Design Guide* for more information about global variables.

`$_processContext` contains general information about the process, such as the process ID, the project name,whether the process was restarted from a checkpoint, and so on.

Only global variables that have the Deployment option checked (on the advanced editor dialog) are visible in the `$_globalVariables` process variable. Also, only global variables with well-formed XML names (for example, names containing a % are not well-formed) appear in the `$_globalVariables` process variable.

Error Process Variables

TIBCO ActiveMatrix BusinessWorks provides error process variables to contain data pertaining to the error that has occurred. TIBCO ActiveMatrix BusinessWorks BPEL Extension does not have error transitions like TIBCO ActiveMatrix BusinessWorks. Instead, faults must be caught and handled by exception-handling routines. Because a fault contains the information within the error process variable, this type of variable is not as useful to an orchestration process.

See Chapter 6, Exception Handling, on page 85 for more information about handling errors.

Memory Usage of Process Variables

All process variables in a running process instance are stored in memory and therefore consume system resources. When variables are no longer referenced by activities in an orchestration process, they are cleaned from memory to free memory for other processing.

The `EnableMemorySavingMode` property controls whether memory for process variables is cleaned up on a per-process instance or per-engine basis. See *TIBCO ActiveMatrix BusinessWorks Administration* for more information about setting properties.

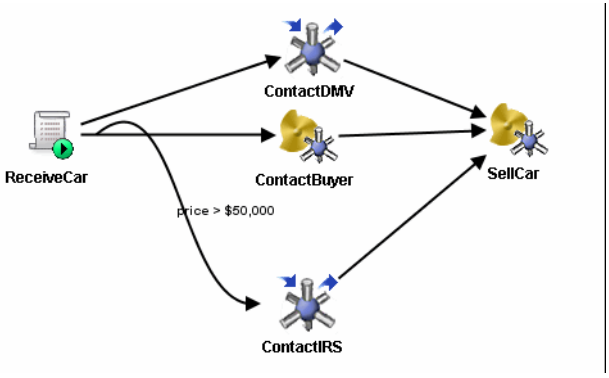
Transitions and Join Conditions

Transitions in orchestration processes are somewhat similar to transitions in TIBCO ActiveMatrix BusinessWorks process definitions. That is, transitions between activities indicate the order that the activities should be processed. Also, conditions can be placed on the transitions between activities to determine if the transition should evaluate to true or false.

The BPEL specification describes links between activities. Links are synonymous with transitions in this manual. Because this product is an extension of TIBCO ActiveMatrix BusinessWorks, the term transition is used to be more compatible with TIBCO ActiveMatrix BusinessWorks terminology.

Figure 24 illustrates a portion of an orchestration process that uses transitions and conditions to control the sequential processing of events. A new car is received and then three transitions are drawn to the next activities in the sequence. Two activities, ContactDMV and ContactBuyer are executed without condition. A third activity, ContactIRS is executed only if the price of the car is greater than $50,000 because a special tax may need to be reported upon the sale of the vehicle.

Figure 24 Auto sales example

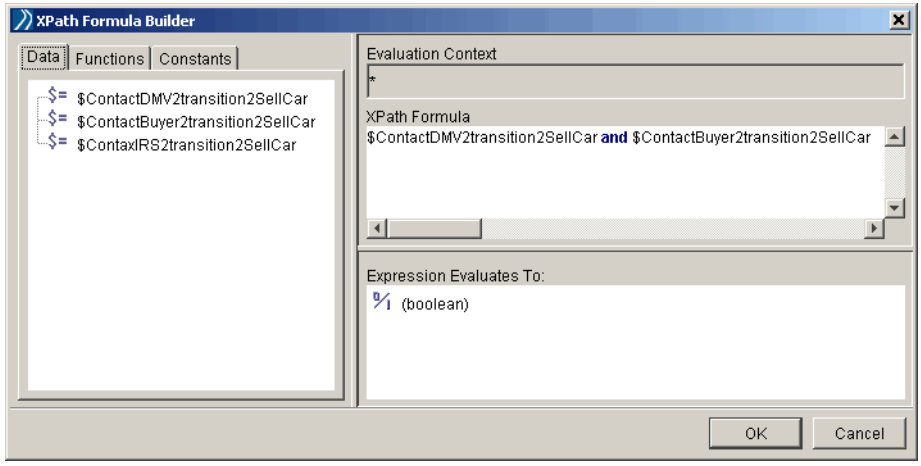


By default, if at least one transition leading to an activity evaluates to true, the activity executes. Also, if all transitions that lead to an activity evaluate to false, the activity does not execute. In the example in Figure 24, the SellCar activity will execute as long as one of the previous three activities executed successfully.

Orchestration processes let you override the default flow of processing. You can define join conditions that evaluate the incoming transitions to any activity. The expression of the join condition can evaluate to true or false. If true, the activity executes, if false, the activity can be skipped or a joinFailure exception can be thrown.

Join condition expressions are specified on the Join Condition tab of the activity. You can use the XPath Formula Builder to drag and drop values and operators to create the expression. Only the state of the incoming transitions to an activity can be used to define a join expression. In the car purchasing example in Figure 24, you may want to specify a join condition on the SellCar activity so that it executes only if both the ContactDMV and ContactBuyer activities execute. Figure 25 illustrates using the XPath Formula Builder to create the join expression.

Figure 25 Using the XPath Formula Builder to create a join condition expression



The default action to perform if the join condition evaluates to false is specified by the current scope. That is, on each group, you can specify an action to perform, or the group inherits the action of its parent scope. The Default Join Action field on the Configuration tab of the Orchestration Process resource specifies the default join action for the top-level scope.

You can override the default action by specifying the action to perform on the Join Condition tab of each activity. The values of the Join Action field on each activity are the following:

- Inherit setting — use the default action specified within the current scope. The current scope is either the current group containing the activity or the orchestration process if the activity is not within a group.
- Bypass activity if false — if the join expression of the activity evaluates to false, then skip the processing of the activity, set the outgoing transition conditions to false without evaluating them, and resume processing.
- Throw fault if false — if the join expression of the activity evaluates to false, throw a joinFailure fault. See Chapter 6, Exception Handling, on page 85 for more information about handling faults.

Correlations

Correlations are used to ensure that messages are delivered to the correct running instance of an orchestration process. Activities can set the value of a correlation based on any data contained within a message sent or received by the orchestration process instance. Each activity may specify a different expression against a message that is used to derive the value of a correlation. Other activities that send or receive messages in the orchestration process may specify another expression used to match the correlation value to determine which messages should be routed to running instance of the orchestration processes.

For example, a new order may have an order number. The order number can then be used by all messages that apply to that particular order. A buyer, for example, can add items to an existing order by sending a message with the correct order number. The activities in the orchestration process would set a correlation to the value of the order number for incoming messages. Messages with the same order number would be automatically routed to the correct orchestration process.

Correlations are defined in the Orchestration Process resource and then used by the activities that send or receive messages, such as the following:

- Receive Starter
- Invoke
- Invoke Process
- Receive
- Reply
- Reply With Fault
- On Event

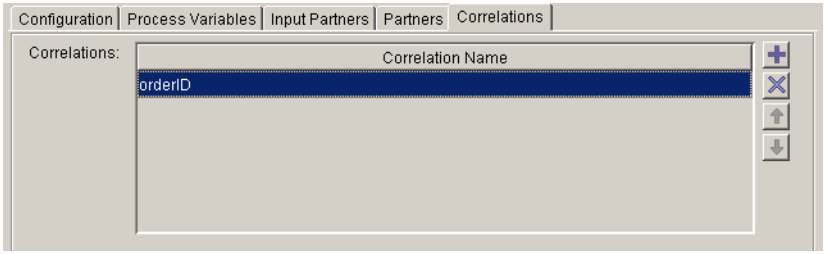
You can define more than one correlation for each activity that handles incoming or outgoing messages. When comparing correlations of incoming messages, comparisons for all defined correlations must evaluate to true for the message to be routed to the receiving activity. For example, you may have separate correlations defined for the orderID of a message and a particular userID that requested the order. Only the original requestor can request the status of the order. In this case, both the orderID correlation and the userID of the requestor must match in the incoming message.

To use correlations, perform the following:

1. On the Orchestration Process resource, click the Correlations tab.

2. Use the + button to add correlations to the process, use the X button to remove correlations, and use the arrow buttons to move the correlations in the list.
3. Double click on the correlation name to specify the name of the correlation.

   The following illustrates the Correlations tab defining a correlation named orderID:



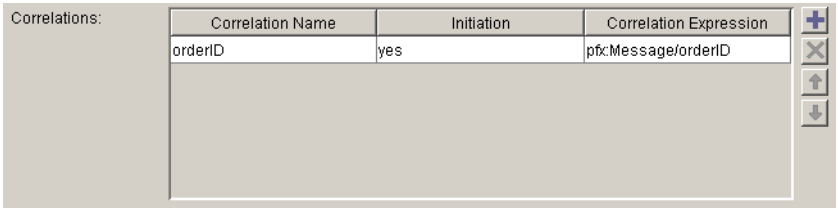
4. Select the Receive Starter activity of the process.
5. Use the + and X buttons in the Correlations field on the Configuration tab to add or delete correlations that you wish to set for the incoming message. Double click on the correlation name and use the drop-down list to select one of the correlations defined in step 3.
6. In the Initiation column select one of the following options from the list:

   — **yes** Use this option if you wish to set the correlation to a value from the incoming message. The specified correlation will then be set to the value determined by the Correlation Expression field.

   — **join** Use this option if you have more than one Receive Starter activity for the process. This option signifies that if the specified correlation is not set, then set it to the value determined by the Correlation Expression field. If the specified correlation is already set, let the current value of the correlation remain unchanged. This option is useful when more than one message can start an orchestration process and you want the first received message to set the correlation value. Other incoming messages with the same correlation value are delivered to the other Receive Starters in the process instead of creating new process instances.
7. Click on the far right corner of the Correlation Expression field and the XPath Formula Builder appears. Use this dialog to create an expression from the data of the incoming message. The correlation is set to the evaluation of this expression. For example, if your incoming message has a field named orderID, you can drag and drop it into the XPath formula field to set the correlation to the value of the orderID field of the incoming message.

   See *TIBCO ActiveMatrix BusinessWorks Process Design Guide* for more information about using the XPath Formula Builder.

The following illustrates a Receive Starter Correlations field that sets the orderID correlation created in step 3 to the value of the orderID field of an incoming message:



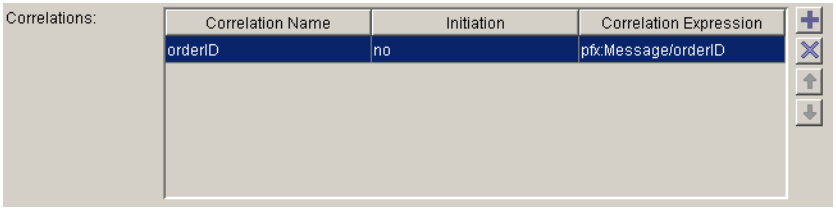
8. Repeat steps 5 through 7 for other activities in the process that send or receive messages. In the Initiation column of all activities (except the Receive Starter), you can set one of the following values:

   — **yes** Use this option if you wish to set the correlation to a value from the incoming message. The specified correlation will then be set to the value determined by the Correlation Expression field. If the correlation already has a value, an exception is thrown.

   — **no** Use this option if you wish to compare the current value of a correlation to the value of the Correlation Expression column (derived from the incoming message). This is useful if you wish to ensure that messages with the same value for the Correlation Expression are handled within the same instance of the orchestration process.

   — **join** This option signifies that if the specified correlation is not set, then set it to the value determined by the Correlation Expression field. If the specified correlation is already set, let the current value of the correlation remain unchanged. This is useful if more than one activity can potentially set a correlation after a process starts and it is unknown which activity will be executed first.

   The following illustrates the Correlations field of an On Event activity that is waiting for a message that has the same orderID as the message that started the orchestration process:



Invoking Services From TIBCO ActiveMatrix BusinessWorks Process Definitions

TIBCO ActiveMatrix BusinessWorks process definitions can invoke ActiveMatrix BusinessWorks BPEL Extension services by using the Invoke Service activity within a process definition. The TIBCO ActiveMatrix BusinessWorks process definition acts as a client of the service and invokes the specified operation.

You can select Service resources stored within the same TIBCO ActiveMatrix BusinessWorks project, or you can select WSDL resources with concrete bindings to external services stored within the project. However, when invoking a service in the same project, only local endpoints are exposed.

SOAP endpoints defined on services within the same project as a TIBCO ActiveMatrix BusinessWorks process definition are not listed in the Endpoint field of an Invoke Service activity. If you wish to invoke a local service by way of a SOAP endpoint, save the WSDL file of the service in the project directory, refresh the project to load the new WSDL, then use a SOAP Request/Reply activity to invoke the service.

See Invoke Service on page 132 for more information about configuring the Invoke Service activity.

Chapter 5

Scopes, Iteration, and Conditional Processing Using Groups

The WS-BPEL specification describes the use of structured activities for iteration or conditional processing. Also, the WS-BPEL specification describes the use of scopes to create sets of activities that have common variables, correlations, error handling routines, and so on. TIBCO ActiveMatrix BusinessWorks uses groups for all of these purposes.

This chapter describes the use of groups in the TIBCO ActiveMatrix BusinessWorks BPEL Extension.

Topics

Overview of Groups

Groups are used in TIBCO ActiveMatrix BusinessWorks to specify related sets of activities. A WS-BPEL scope is roughly analogous to a TIBCO ActiveMatrix BusinessWorks group, except a scope has the capability to define variables and correlations for use within the scope. Also, many of the types of groups in TIBCO ActiveMatrix BusinessWorks are similar to WS-BPEL structured activities.

The TIBCO ActiveMatrix BusinessWorks BPEL Extension uses the graphical representation of groups to provide the functionality of WS-BPEL scopes and the conditional or iterative processing of WS-BPEL structured activities. The following types of groups are available:

- Scope — similar to a scope or a flow in the WS-BPEL standard. This type of group allows you to specify a set of activities that have a common set of variables, correlations, and error-handling routines.
- Iterate — similar to a repeat until or for each structured activity in the WS-BPEL standard. This type of group allows you to repeat a set of activities once for each value in a list.
- Pick First — similar to a pick structured activity in the WS-BPEL standard. This type of group allows you to specify that the first activity that completes should determine which transition to take to continue processing. You can use this type of group to wait for one or more incoming events and continue processing based on which event is received first.
- While True — similar to a while structured activity in the WS-BPEL standard. This type of group allows you to repeat a set of activities as long as a condition is true.
- If — similar to an if structured activity in the WS-BPEL standard. This type of group allows you to specify that a set of activities is to be executed conditionally, such as in an `if ... then ... else if ...` construct in a programming language.

Activities can be grouped or ungrouped. Also, groups can be maximized to display all activities in the group or minimized to show only a small icon for the whole group. This allows you to collapse and expand groups in an orchestration process to better display the relevant portions of the process you wish to view. Maximized groups can also be resized.

To group a set of activities:

1. Choose the Select tool (the arrow pointer in the tool bar).
2. In the design panel, draw a box around the desired activities.

3. Choose **View>Create a Group** from the menu, or click the Create a group  icon.

   The group configuration appears in the configuration panel.

4. Specify the type of group to create and any other configuration parameters required for the group. See Table 7 for more information about the fields of the group configuration tab.

5. Draw a transition from the start of the group to the first activity to execute in the group. The start of the group is the green start arrow on the left side of the group box.

6. Draw a transition from the last activity to execute in the group to the end of the group. The end of the group is the red end square on the right side of the group box.

To ungroup a set of grouped activities:

1. Choose the Select tool (the arrow pointer in the tool bar).

2. Select the group in the design panel.

3. Choose **View>Remove a Group** from the menu, or click the Undo the group button.

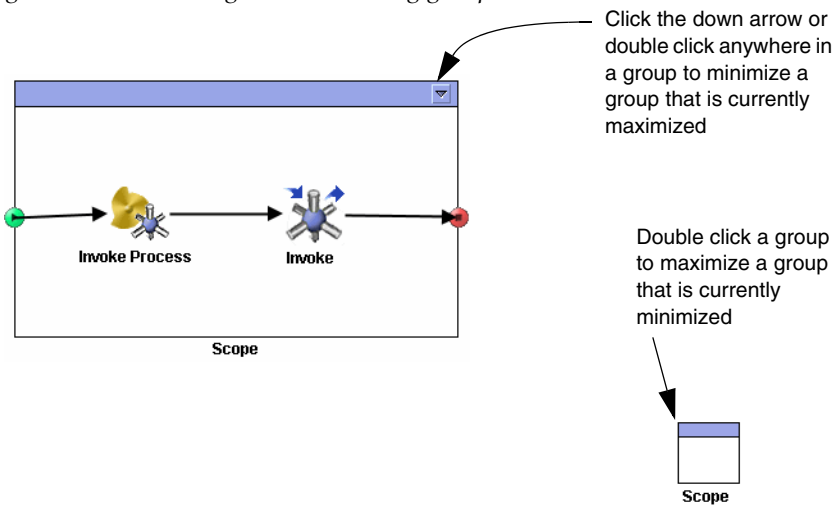
To minimize or maximize the display of a group:

1. For groups that are currently maximized, click the down arrow in the upper right-hand corner of the group or double click anywhere in the group to minimize the group.

2. For groups that are currently minimized, double click the group icon to maximize the group.

   Figure 26 illustrates minimizing and maximizing a group.

Figure 26 Minimizing and maximizing groups



To resize a maximized group:

1. Maximize the group, if it is not already maximized.
2. Choose the Select tool (the arrow pointer in the toolbar) and select the group in the process definition.
3. Click and drag the desired anchor point on any side or on the corners until the group is the desired size.

Activity Output and Groups

Each activity in the group can access the output of previously set variables inside or outside the group. If the group is used for a loop (iterate or while true), the value of all variables declared in the group (including activity output) is reset so that activities in subsequent iterations of the group will not have access to output data from previous iterations. Also, any loop indexes for loops contained in loops are reset when the parent loop begins a new iteration.

When a group has completed executing, output from the activities in the group is available to subsequent activities in the orchestration process. In the case of loop groups, only output from the last execution of the activity is available.

Configuring Groups

All groups have the following tabs:

- Configuration Tab
- Group Variables Tab
- Correlations Tab
- Join Condition Tab

The following sections describe the options available on each tab.

Configuration Tab

The Configuration tab is used to specify the type of group and its configuration. Table 7 describes the fields in the Configuration tab for groups.

Table 7 Group Configuration tab

Field	Description
Name	The name to appear as the label for the group in the orchestration process.
Description	Short description of the group.
Isolated	When checked, the group becomes an isolated scope. Isolated scopes cannot execute concurrently with any other isolated scopes within the orchestration process. Isolated scopes are executed in the order in which they are reached in the orchestration process. **Note:** Nested isolated scopes are flagged as an error when the orchestration process is validated. Do not create nested isolated scopes.

Table 7 Group Configuration tab

Field	Description
Group Action	The type of group. Groups can be of the following types: • Scope — See Scope Groups on page 75 for more information. • Iterate Loop — See Iterate Loop on page 80 for more information. • Pick First — See Pick First Groups on page 77 for more information. • While True Groups — See While True Loop on page 83 for more information. • If Groups — See If Groups on page 76 for more information.
Group Action: Iterate	
Index Name	The index variable for the loop. This variable is used to store the current iteration number of the loop. The index starts at one and increments by one with each execution of the loop. See Index Variable on page 79 for more information.
Variable List	A process variable (or XPath expression) containing the list you wish to use as the source of the iterations. The group iterates once for each item in the list. Use the button to the right of the field to create an XPath expression for this field.
Iteration Element	A name to use for the process variable containing the current iteration element of the data supplied in the Variable List field. See Iteration Element on page 81 for more information on this field.
Accumulate Output	Specifies that you wish to accumulate the output of each execution of one of the activities in the group into a process variable. See Accumulate Output on page 79 for more information.

Table 7 Group Configuration tab

Field	Description
Output Activity	The activity in a group for which you wish to accumulate output for each execution of the loop. You may select only one activity in the group.
Output Name	The name of the process variable to store the successive output of the selected activity in the Output Activity field.
Group Action: While True	
Index Name	The index variable for the loop. This variable is used to store the current iteration number of the loop. The index starts at one and increments by one with each execution of the loop. Specifying an index name is optional for While True loops. See Index Variable on page 79 for more information.
Conditions	The loop repeats as long as the condition specified in this field evaluates to true. The condition is evaluated when the group is entered. If the condition evaluates to false, the activities in the group are not executed. The condition is specified as an XPath expression and the XPath Formula Builder is available to help to create the condition. See *TIBCO ActiveMatrix BusinessWorks Process Design Guide* for more information.
Accumulate Output	Specifies that you wish to accumulate the output of each execution of one of the activities in the group into a process variable. See Accumulate Output on page 79 for more information.
Output Activity	The activity in a group for which you wish to accumulate output for each execution of the loop. You may select only one activity in the group.
Output Name	The name of the process variable to store the successive output of the selected activity in the Output Activity field.

Group Variables Tab

The Group Variables tab allows you to create process variables that are available only within the scope of the group. See Process Variables on page 55 for more information about process variables.

Correlations Tab

The Correlations tab allows you to define named correlations that are available only within the scope of the group. See Correlations on page 62 for more information about correlations.

Correlation names must be unique for all scopes within the orchestration process. If you specify a correlation name within a scope that has the same name as a correlation in a parent scope, you will receive a validation error when you validate the project before deployment.

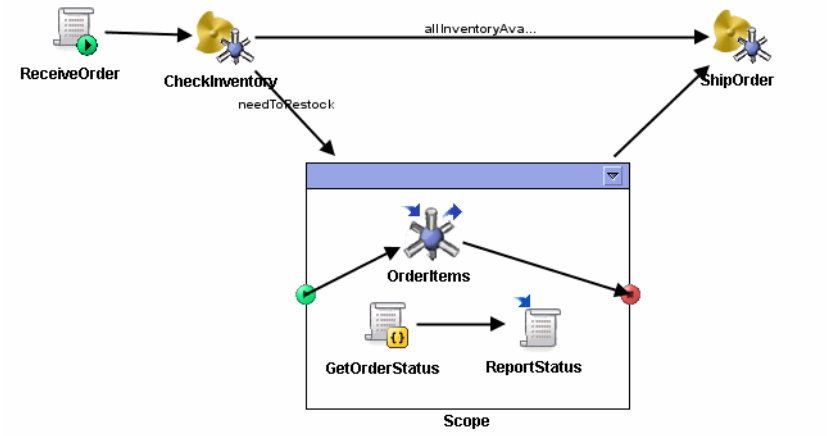
Join Condition Tab

The Join Condition tab allows you to specify join expression for the group as well as the default join action for activities within the group. See Transitions and Join Conditions on page 60 for more information about join conditions.

Scope Groups

A group with the Group Action field specified as Scope is similar to a WS-BPEL scope. You can define variables or correlations that apply only to activities within the group. You can also define error handling and event handling routines that execute only within the scope of the group.

The following orchestration process illustrates a scope group.

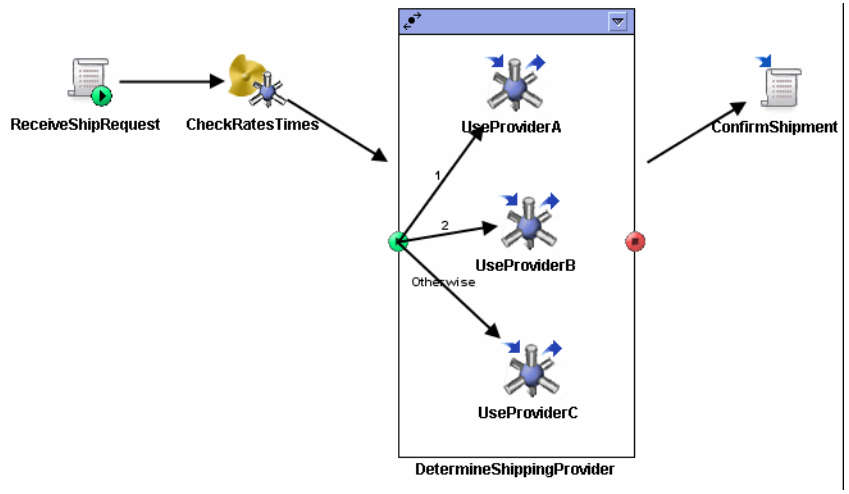


The process performs the following operations:

1. A new order is received.
2. A check is performed to determine if the order can be fulfilled with existing inventory. If it can, the order is shipped. If there is not sufficient inventory, the group named Scope is executed.
3. In the group named Scope, an Invoke operation is performed to order the necessary items from an outside vendor.
4. While the group Scope is executing, the requestor of the order can send another message asking for the status of the order.
5. Once the items have been ordered, the order is shipped.

If Groups

You can use groups to conditionally execute business logic. The If group allows you to specify a set of conditions that are evaluated in order. The following is an example of conditional execution using an If group.



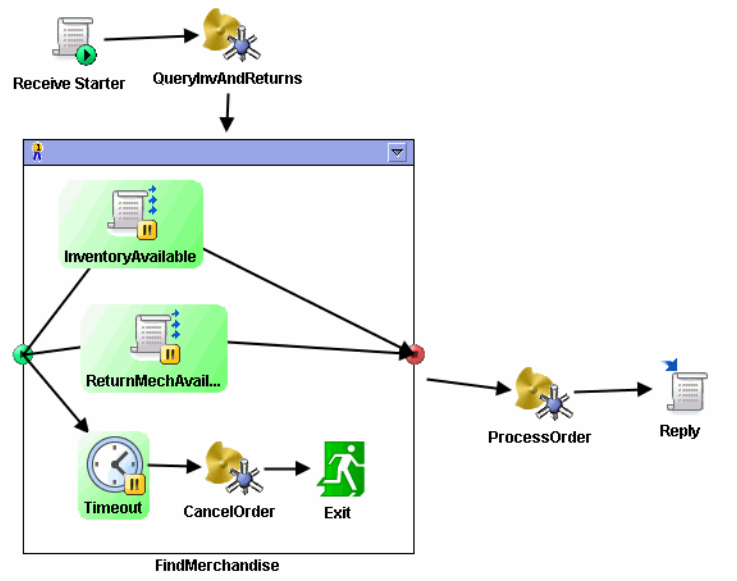
The transitions from the start of the If group specify the conditions to be evaluated. Each transition is numbered and evaluated in order. The first condition to evaluate to true determines which execution path to take in the group. All other conditions are ignored once one evaluates to true. An "otherwise" condition can be specified to handle the case when no other conditions evaluate to true. If no conditions evaluate to true, and no otherwise condition is specified, the activities in the group are not executed and processing continues with the first transition after the group.

The example If group illustrates an orchestration process that receives a shipping request. A check is made to determine the current shipping rates and timetables for the vendors. A formula is used in the conditions to evaluate each vendor's price to shipping time ratio with the corporate standard. The first vendor whose ratio evaluates to lower than the corporate standard is used.

Pick First Groups

Pick first groups allow process execution to wait for one or more events. The first event that completes determines which transition to take to continue processing. For example, as part of an order-entry system, when an order is placed, a check is made to see if the order can be filled from stocked inventory or from returned merchandise. Whichever system returns the information first is used to fill the order. If neither system returns the information about available inventory, the order times out and cancels.

The following illustrates an orchestration process that uses the Pick First group to implement the business logic described above. An order is received and a process is called to query the inventory system and the returned merchandise system. A transition is then taken to the Pick First group. The group then waits for either the return message from InventoryAvailable, the return message from ReturnMerchAvailable, or the Timeout activity. The first activity to complete determines the next transition taken. If either InventoryAvailable or ReturnMerchAvailable complete first, the transition to the ProcessOrder activity is taken. If the Timeout activity completes first, the transition to the CancelOrder activity then the Exit activity is taken.



To specify the events that you would like to wait for, draw transition lines from the start of the group to the desired activities. Only Receive Starter, Receive, and Sleep activities can have valid transitions from the start of the Pick First group. If the transition is valid, the activity is highlighted in green. If the transition from the start of the group is drawn to an invalid activity, the activity is highlighted in red.

Overview of Loops

Loops allow you to execute a series of activities more than once. You can iterate based on the items in an array stored in a process variable or you can iterate while a given condition is true. The following are the types of loops that are available:

- Iterate Loop
- While True Loop

Loops allow you to accumulate the output of a single activity in the loop for each execution of the loop. This allows you to retrieve output from each execution of the activity in the loop. See Accumulate Output on page 79 for more information about accumulating the output of each iteration of a loop.

Index Variable

The index variable holds the current number of times a loop has executed. The iteration count starts at one the first time the loop is executed, and the count increases by one for each iteration of the loop. You can access this variable like any other process variable by referencing it with a dollar sign (`$`) in front of it.

For nested loops, the index of the contained loop resets at the beginning of each iteration of the parent loop.

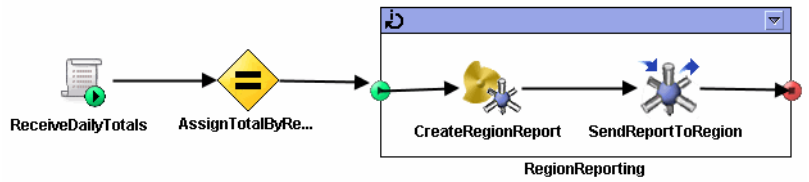
Accumulate Output

You can accumulate the output of one of the activities in a group by checking the Accumulate Output field. If you check this field, you can select one of the activities in the group, and each time the loop is executed, the selected activity's output is placed into a list. The list of accumulated output for that activity is stored in a variable whose name is specified in the Output Name field. After the loop exits, this variable can be accessed in the same way other process data can be accessed by other activities.

Because you can accumulate output from only one activity in a group, you should design your group so that only one activity in the group holds the data to accumulate for each iteration. For example, you may wish to accumulate the sum of the amount for line items in an order.

Iterate Loop

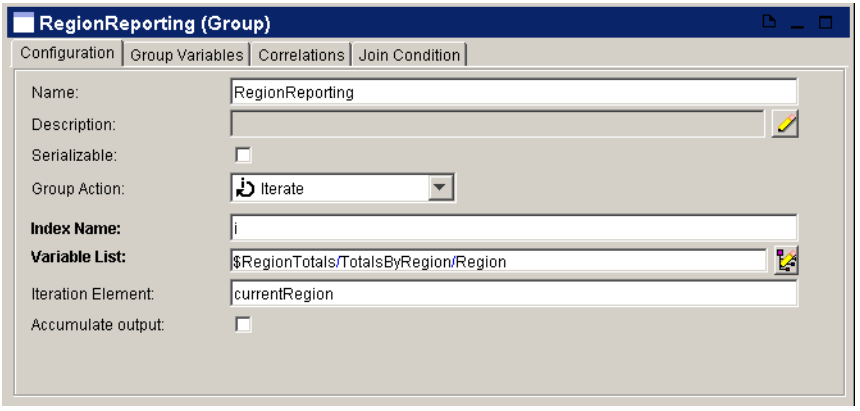
An Iterate loop repeats the series of grouped activities once for every item in a list. The list is stored in a process variable and can be items of any datatype. The following is an example of an iterate loop.



The orchestration process performs the following operations:

1. A message is received containing the daily totals of products sold in each region.
2. The Assign activity assigns each region's total to a process variable containing a list of each region.
3. The group iterates over each region stored in the process variable and performs the following:
   d. Creates a report based on the totals reported for the region.
   e. Sends the report to the service that distributes the information to the appropriate offices in each region.

The following is the configuration for this example:



In this example, the process variable `$RegionTotals` is used to determine the number of iterations to perform. One iteration is performed for every element contained in the repeating element.

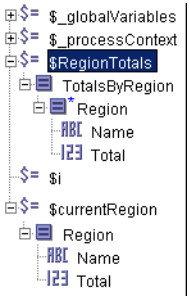
The Variable List field is an XPath expression. You can use a simple expression containing a complete list as in the example above, or your expression can be more complex and only process certain items in the list. For example, if you wish to skip over the first 10 regions, the expression in the Variable List field would be the following:

```
$RegionTotals[position() > 10]
```

See *TIBCO ActiveMatrix BusinessWorks Process Design Guide* for more information on creating XPath expressions.

Iteration Element

The Iteration Element field on the Configuration tab allows you to supply a name for a process variable containing the current iteration element. This allows you to easily map the value of the current iteration element instead of using predicates on the process variable used for iteration. When you specify a value for this field, a process variable with the specified name appears in the Process Data tree in the Input tab. For example, in the Iterate group above, we specified `currentRegion` as the name for the current element. This causes the following to appear in the process data tree:

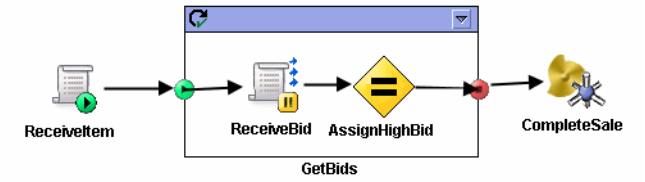


Notice that both `$RegionTotals` and `$currentRegion` contain the element `Region`. `$currentRegion/Region` is a copy of the `$RegionTotals/TotalsByRegion/Region[i]` element, the current element being processed. If you use `$RegionTotals/TotalsByRegion/Region[i]` in an input mapping, TIBCO ActiveMatrix BusinessWorks BPEL Extension traverses the `$RegionTotals/TotalsByRegion/Region` element to retrieve the current element each time the loop iterates. By using the `$currentRegion/Region` element instead, you save processing time in proportion to how many times the

iteration loop repeats. The larger number of elements contained in the `$RegionTotals/TotalsByRegion/Region` repeating element, the greater the performance improvement you will notice by using `$currentRegion/Region` instead of `$RegionTotals/TotalsByRegion/Region[i]`.

While True Loop

The While True loop repeats the series of grouped activities as long as the given condition evaluates as true. The condition is evaluated when the group is entered. If the condition evaluates to false, the activities within the group are not executed. The following is an example of a While True loop.



The process performs the following operations:

1. A clearing house application receives a message containing the description of an item up for sale and the price the seller is willing to accept.
2. The While True group is executed. The condition of the group is that the current high bid has to be greater than the price the seller is willing to accept.
3. Bids are received and each higher bid sets the high bid process variable.
4. When the high bid process variable is greater than the seller's price, the transition is taken to the CompleteSale activity.

Chapter 6 Exception Handling

This chapter describes the error-handling functionality of TIBCO ActiveMatrix BusinessWorks BPEL Extension.

Topics

Overview of Exception Handling

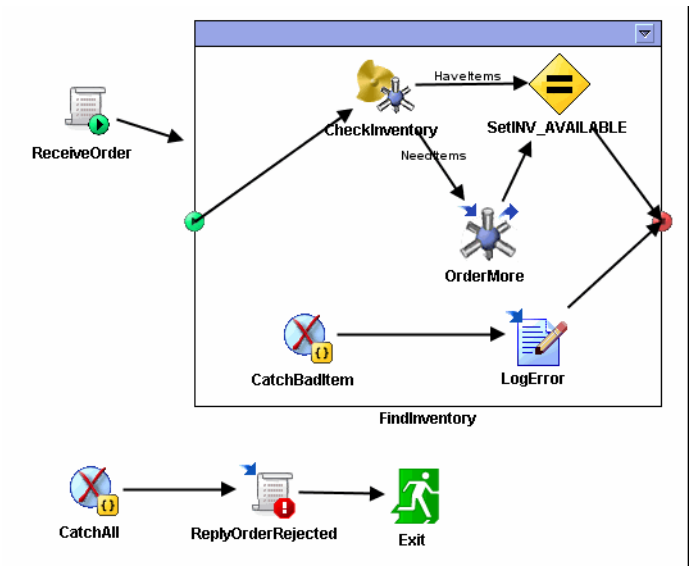
Orchestration processes can encounter errors during processing, such as services being unavailable or improperly formatted incoming messages. TIBCO ActiveMatrix BusinessWorks BPEL Extension allows you to catch errors and determine what actions to perform in the event of an error.

Potential errors, also known as exceptions or faults, that each activity can raise are displayed on the Error Output tab of each activity. See Chapter 10, Palette Reference, on page 115 for more information about each activity's potential errors.

The Catch activity is used to trap errors and define an error handling routine. This activity can catch specific errors that activities can encounter, or the activity can be configured to catch all errors that do not have their own error-handling routine. Processing transfers to the appropriate Catch activity when an error occurs, and then you can create transitions to other activities to perform actions to handle the error. You cannot draw transitions to the Catch activity or from activities in the error handling routine to the main processing flow. The only transitions that are permitted are transitions to the end of the scope.

Figure 27 illustrates a partial orchestration process that has two error handling routines.

Figure 27 Example of error-handling



The CatchAll activity begins the error-handling routine defined at the top-level scope for the process. This routine catches any unhandled error, replies with a fault message, then exits the process.

The group FindInventory is a scope where the inventory is either available in current stock or must be ordered from an outside vendor. Processing within the scope begins with an activity that checks the status of inventory for the items ordered. If inventory is available, the INV_AVAILABLE process variable is set to true and the scope exits. If inventory is not available, an invocation to an external service orders more inventory, then the variable is set and the process continues to execute. Within this scope, if an invalid item is requested, the BadItem error is raised and processing transfers to the CatchBadItem error-handling routine. In the event of an error, the error is written to the log file then processing transitions to the end of the scope.

Errors that occur that are not handled by any error-handling routine are propagated to the calling environment. In the example in Figure 27, the CatchBadItem activity catches any BadItem errors that occur while the FindInventory group is executing. The CatchAll activity catches any other error.

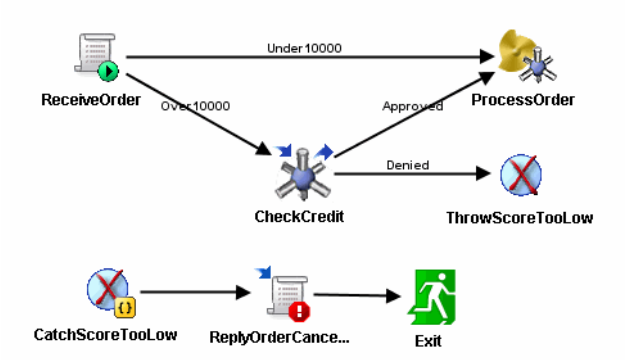
Errors can also occur during the processing of an error-handling routine. In this case, the error cannot be caught by any error-handling routine within the same scope. The error is propagated to the parent scope.

Throw Activity

The Throw activity allows you to throw an exception. The exception can be a potential exception for any of the activities in the orchestration process, or you can define a new exception in the Throw activity.

The Throw activity is particularly useful if an error in business logic occurs during processing. For example, orders over $10,000 must include a credit check of the customer. If the credit check returns a credit score that is too low, the order is canceled. The Throw activity can be used to return a fault message to the order requestor stating that the credit check has failed. Figure 28 illustrates this example.

Figure 28 Using the Throw activity

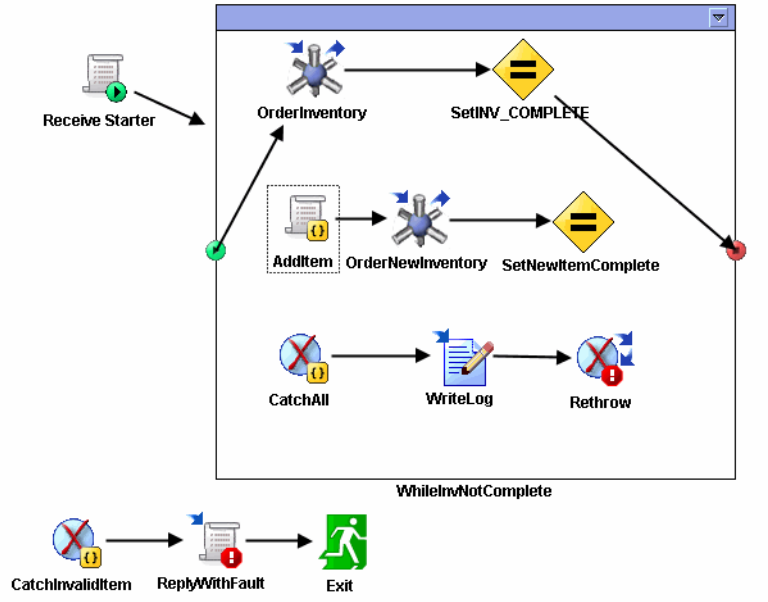


Rethrow Activity

The Rethrow activity is used within an exception-handling routine to throw the caught error again. This is useful if you wish to perform some processing within an error-handling routine, but you also wish to send the error to the next higher scope.

For example, an order-entry process may accept new items as long as the current order is still in the process of ordering the items from the vendor. A group is used to contain the activities that order the items from the vendor. If an error occurs during that time, a message is written to the log then the error is rethrows to the main orchestration process. The main orchestration process catches the invalid item error and sends a fault message to the order requestor and exits the process. Figure 29 illustrates this example.

Figure 29 Using the Rethrow activity



Chapter 7 Services and Partners

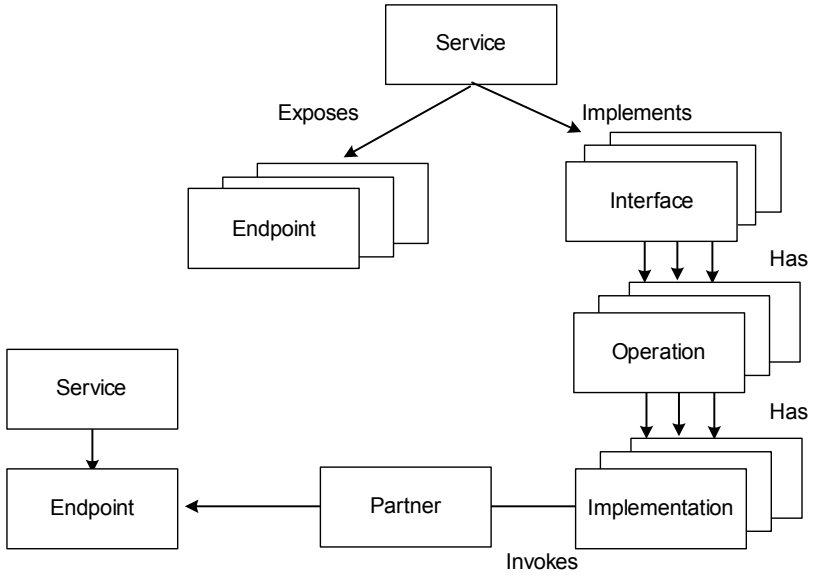
This chapter describes how to configure services and partners in your TIBCO ActiveMatrix BusinessWorks BPEL Extension project.

Topics

Overview of Services and Partners

Services are the basic unit of operation within a service-oriented architecture (SOA). A service publishes one or more interfaces that other applications and services can use to invoke the operations contained within the service. The service exposes its interfaces through endpoints, such as HTTP or SOAP. The implementation of the operations in the service can also call other services, or partners. Figure 30 illustrates the service model.

Figure 30 Service model



Services decouple the underlying transport from the implementation. You can define an interface you wish to offer, describe the endpoints through which clients can access the service, and specify the implementation for each operation in the service. The underlying implementation of the operations can change without affecting the configuration of the service.

Also, partner link configurations allow you to decouple partners from their endpoints. You can specify the type of the partner you wish to invoke, and a Partner Link Configuration resource associates the partner name with the endpoint of the partner service you wish to invoke. Partner Link Configuration resources allow you to provide endpoints to new partner services when you want to switch services without changing the orchestration process that invokes the partner service.

Configuring a Service

Use the Service resource to configure services for your project. Before configuring a service, you will need to have the following in your project:

- WSDL resources for the interfaces the service publishes. You can use abstract WSDL files or WSDL files with concrete endpoint bindings. However, concrete bindings are ignored because the Service will define its own endpoint bindings. You might use a WSDL file with concrete bindings if you are modeling your service on the WSDL of another existing service.
- WSDL resources for any external partner services that operations can invoke. The WSDL files for partner services must have concrete endpoint bindings so that TIBCO ActiveMatrix BusinessWorks BPEL Extension can invoke the service.
- Orchestration processes that provide the implementation of the operations.
- HTTP Connection and JMS Connection shared configuration resources for the transports used by the endpoints.
- Partner Link Configuration resources for any service partners that your orchestration processes invoke.

Only the in or in-out message exchange pattern is supported in WSDL files used by the ActiveMatrix BusinessWorks BPEL Extension.

See the TIBCO ActiveMatrix BusinessWorks documentation for more information about creating and importing WSDL files into projects and defining HTTP Connection resources. See Working with Orchestration Processes on page 47 for more information about creating orchestration processes. See Partners on page 96 for more information about partners and Partner Link Configuration resources.

To configure a service:

1. Drag and drop a Service resource from the Service palette to the design panel.
2. Specify a name for the service in the Name field, and optionally specify a description for the service in the Description field.
3. Select BPEL Extension in the Implementation Type field, if it is not already selected.
4. Click the + button to the right of the box in the Implementation field to add a WSDL file that describes the interface of the service.
5. In the Select a Resource dialog, locate and select the WSDL resource for the service, and select the correct Port Type for the service.

6. Repeat steps step 4 and step 5 as necessary to add more interfaces to the service, if necessary. Interfaces are added as tabs in the Implementation field. Click on the tab of the interface you wish to configure to continue the configuration. Use the **Rename** button to rename the selected interface, or use the **X** button to delete an interface, if necessary.
7. For each interface of the service, perform the following:
    a. Specify a timeout for each operation listed in the operations tab. This timeout applies to how long incoming messages that are routed to a particular running job wait for the job to process them. Messages are discarded after this timeout has been reached.
    b. Double click on the Operation Implementation field for each operation and use the **Browse** button to locate the orchestration process that implements this operation. Only orchestration processes with input partner portTypes that match the operation's portType are listed. See Input Partners on page 97 for more information about input partners and Chapter 4, Working with Orchestration Processes, on page 47 for more information about creating orchestration processes.
    c. Click the Endpoint Bindings tab and configure one or more endpoints that expose this interface.
    d. Click the + button to add endpoints. Use the X button to remove endpoints. Use the arrow buttons to move the endpoints up or down.
    e. For each endpoint, double click the Endpoint Type field to specify either SOAP or Local. SOAP endpoints allow partners to invoke operations in this service using SOAP over HTTP or JMS. Local endpoints allow only partners contained in the current project to invoke operations within this service.
    f. For SOAP endpoints, specify the HTTP Connection or JMS Connection on the Transport tab, specify any transport configuration options, then specify any desired options on the SOAP Details tab. See Service on page 169 for more information about configuring a SOAP endpoint.
8. If the orchestration implementations within the service invoke partners, the Partner Binding tab allows you to specify Partner Link Configuration resources for the partners. See Partners on page 96 for more information.
9. Once the service is configured, you can view the WSDL for the service on the WSDL Source tab. You can also use this tab to save a WSDL file with the concrete bindings needed for other services or web clients to invoke operations on this service.

Saving Concrete WSDL Files for Partners

TIBCO ActiveMatrix BusinessWorks BPEL Extension allows you save a concrete WSDL file from the Service resource. This WSDL can be used by partner services to call your service. When you save your WSDL file, make certain that the Interface checkbox in the Embed in WSDL field is unchecked (see Figure 31). Checking this box embeds the portType of the interface using a separate namespace, and therefore the portType will not match the namespace of the abstract WSDL. When the Interface checkbox is unchecked, ActiveMatrix BusinessWorks BPEL Extension creates a concrete WSDL with a reference to the interface that partner services can use.

Figure 31 Embed in WSDL field on the WSDL Source tab of a service

Partners

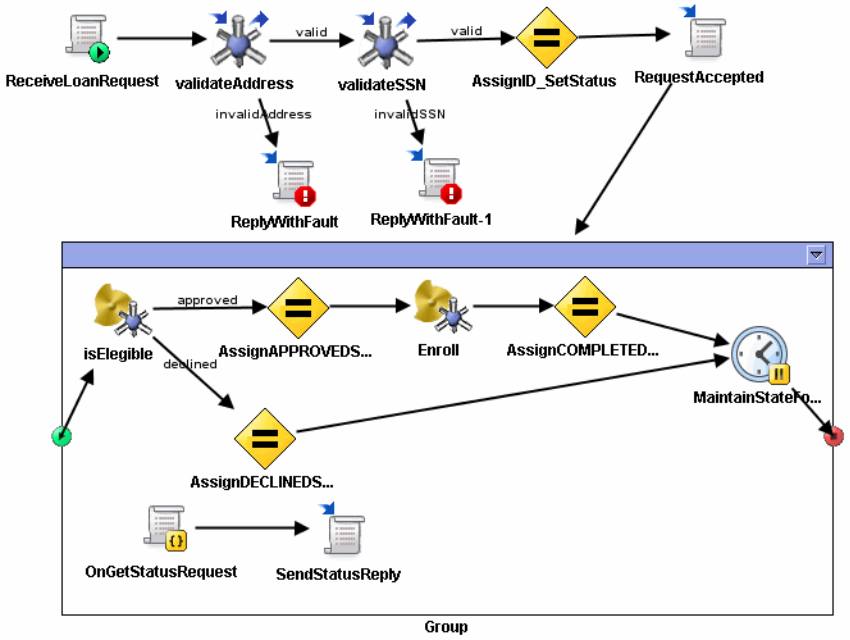
Services that an orchestration process invokes are represented by partners. Partners are defined by a name and a WSDL portType that describes the operations that can be invoked. The Invoke activity is used within an orchestration process to invoke an operation on a partner service.

Services, in turn, can also be called by other services. Therefore orchestration processes declare input partners that describe the various portTypes that are implemented within the orchestration process.

The orchestration process has an Input Partners tab and a Partners tab that allows you to define the partners of the service.

To describe partners and how they are used within orchestration processes, Figure 32 illustrates an online loan application orchestration process.

Figure 32 Online loan application process



In this example, loan requests are received and processed, and applicants can also request the status of the loan while it is still being processed. This process implements the ApplyForLoan and GetLoanApplicationStatus operations of the Loan portType in the following incomplete abstract WSDL:

```
<?xml version="1.0" encoding="UTF-8"?>
...
    <portType name="Loan">
        <operation name="ApplyForLoan"/>
        <operation name="GetLoanApplicationStatus"/>
    </portType>
</definitions>
```

This orchestration process also invokes a partner service for validating the applicant's address and social security number. The following partial WSDL file describes the Validate portType of the partner service.

```
<?xml version="1.0" encoding="UTF-8"?>
...
<portType name="Validate">
        <operation name="isValidAddress"/>
        <operation name="isValidSSN"/>
    </portType>
...

</definitions>
```

Partner WSDL files cannot have imbedded interfaces.

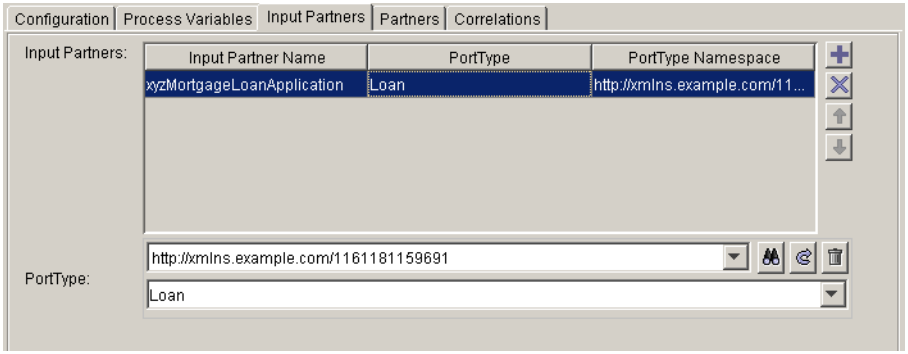
The following sections describe how to configure input partners and partners for orchestration processes.

Input Partners

The Input Partners tab of an orchestration service allows you to specify the various portTypes that are implemented within the orchestration process. Each portType can contain one or more operation, and the orchestration process can then include activities that receive messages for operations.

Consider the online loan application orchestration process example in Figure 32. The orchestration process implements both the ApplyForLoan and GetLoanApplicationStatus operations of the Loan portType. Therefore, the orchestration process has one input partner (the Loan portType). Figure 33 illustrates the configuration of the Input Partners tab of this orchestration process.

Figure 33 Input partner configuration



To configure input partners:

1. Use the **+** and **X** buttons to add or delete partners in the table in the Input Partners field. Move the input partners up or down the list using the arrow buttons.
2. Double click on the value in the Input Partner column of the table to set the name of the input partner.
3. Select an input partner, then use the **Browse** button in the PortType field to locate the appropriate WSDL containing the portType for the input partner.
4. In the Select a Resource dialog, select the appropriate WSDL file and PortType, then click **OK**.

Invoked Partners

Orchestration processes can invoke partner services. Partner services can be located either inside the same project as the orchestration process that invokes the service, or the partner can be an external partner service invoked over the internet by way of the SOAP protocol.

Partner Link Configuration resources associate partner services with their actual endpoints. Partner Link Configuration resources can contain one or more local or external partner services. You use the named partner links within a Partner Link Configuration either within an orchestration process or in the Partner Binding tab of a Service resource.

The general process for configuring partners is the following:

1. Define a Partner Link Configuration resource that specifies the partner's endpoint.
2. Define a partner on the Partner tab of the orchestration process that invokes the partner. The orchestration process requires the port type of the partner.

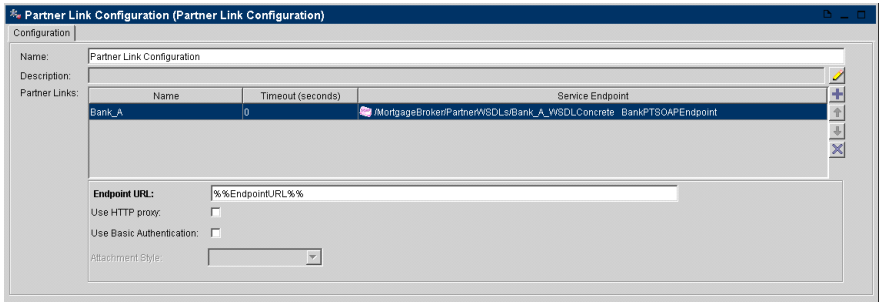
3. Use the partner in one or more Invoke activities to invoke the partner service within the orchestration process.
4. You can bind the partner to its endpoint by doing either of the following:
   a. Specify the Partner Link Configuration resource containing the partner's endpoint on the Partners tab of the orchestration process.
   b. Specify the Partner Link Configuration resource containing the partner's endpoint on the Partner Binding tab of the Service that uses the orchestration process as an implementation of its operations.

The following sections describe creating Partner Link Configuration resources and using these resources within orchestration processes or services.

Partner Link Configuration

Partner Link Configuration resources bind a partner service to its specific port. Bindings can be changed without affecting the underlying orchestration process or service so that you can switch to new partners services with the same portType. Figure 34 illustrates the Partner Link Configuration resource.

Figure 34 Partner Link Configuration resource



To create a partner link configuration resource:

1. Drag and drop a Partner Link Configuration resource from the Service palette onto the design panel.
2. Click the **+** button to add a new partner. If necessary, use the **X** button to delete partners or the arrow buttons to move partners up or down the list.
3. In the Select a Service Endpoint dialog, select a valid endpoint. For WSDL files, valid endpoints are configured ports within the WSDL file. For services within the project, only endpoints configured to use LOCAL as the transport can be specified. If you wish to invoke a local service over SOAP, use a concrete WSDL that exposes the service by way of SOAP.

If necessary, you can later change an endpoint for the specified partner by double clicking on the Service Endpoint column of the partner and selecting a new endpoint.

4. The partner appears in the table in the Partner field. Specify a name for the partner in the Name field. Specify a timeout (in seconds) for the amount of time you wish to allow for invokes to wait while a service responds. Zero indicates an unlimited amount of time.

   Remote partners accessed through SOAP can be further configured, if required. See Partner Link Configuration on page 149 for more information about these fields. Local partners whose services are located within the same project require no additional configuration.

5. You can also specify an HTTP endpoint URL for the relevant partner link in the EndPoint URL field.

   When a new partner is defined against a concrete binding, the URL specified in the concrete binding is set as default. You can modify this URL or set it to a global variable which can be configured at deployment time.

Associating a Security Policy

You can associate a security policy with a service endpoint operation defined for a Partner Link in the Partner Link Configuration resource. The association between the security policy and the service endpoint operation is unique.

The service endpoint operation must be based on a SOAP protocol.

Using the service endpoint operation as the security subject ensures consistency with security policy associations based on the service resource. Service resource also exposes the service endpoint operation as the security subject. Exposing partner links as security policy subjects ensures WS-Security support for all the TIBCO ActiveMatrix BusinessWorks and TIBCO ActiveMatrix BusinessWorks BPEL Extension constructs that use partner links for outbound invocations. The TIBCO ActiveMatrix BusinessWorks BPEL Extension Invoke activity and TIBCO ActiveMatrix BusinessWorks Invoke Partner activity have full WS-Security functionality if they are bound using SOAP protocol.

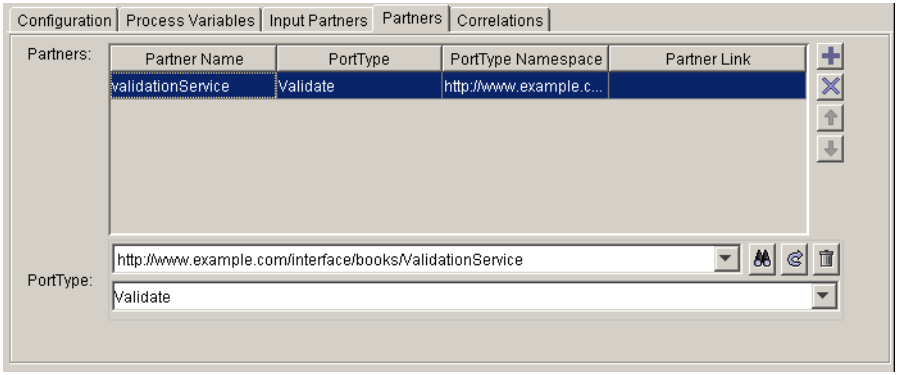
When a security policy is associated with a particular partner link service endpoint operation, the WS-Security processing is enabled and performed according to the WS-Security guidelines for the outbound, inbound, and inbound fault message exchange specified in the Security Policy Association resource.

Security policies in TIBCO ActiveMatrix BusinessWorks BPEL Extension are associated in exactly the same way as in TIBCO ActiveMatrix BusinessWorks. For more information on how to associate a security policy, refer to the TIBCO ActiveMatrix BusinessWorks documentation.

Configuring Partners Within An Orchestration Process

Orchestration processes must define the partner services that are invoked within the process. The Partners tab of the Orchestration Process resource is used to define partners for the orchestration process. Figure 35 illustrates the Partner tab of an orchestration process.

Figure 35 Partner tab on an Orchestration Process resource



To configure a partner service:

1. Use the **+** and **X** buttons to add or delete partners in the table in the Partners field. Move the partners up or down the list using the arrow buttons.
2. Double click on the value in the Partner Name column of the table to set the name of the partner service.
3. Select a partner, then use the **Browse** button in the PortType field to locate the appropriate WSDL containing the portType for the partner.
4. In the Select a Resource dialog, select the appropriate WSDL file and PortType, then click **OK**.
5. Optionally, you can use the Browse button in the Partner Link field to specify the Partner Link Configuration resource that contains the partner's endpoint.

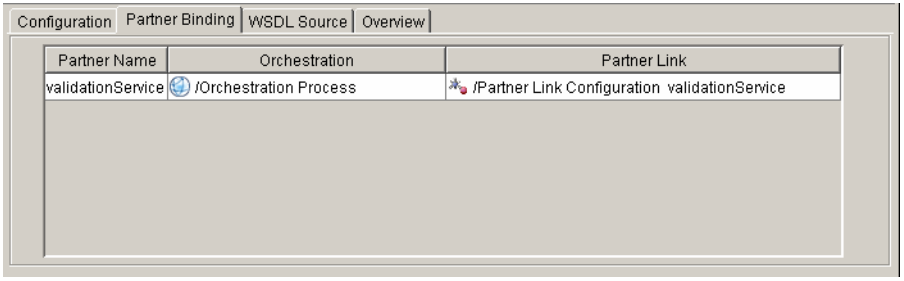
   If you specify the partner link in this field, you do not need to specify the Partner Binding tab in the Service when this orchestration process is used as an operation implementation. If you do not specify the partner link here, you

must specify the partner binding on the service. See Configuring Partner Bindings on a Service on page 102 for more information.

Configuring Partner Bindings on a Service

You can bind partners invoked by all operations in a service on the Partner Binding tab of the Service. Only partners that require binding appear on this tab. For example, if you specified a partner link for the partner on the Partners tab of the orchestration process, the partner does not appear on this tab. Figure 36 illustrates the Partner Binding tab of a service.

Figure 36 Partner Link Configuration resource



To configure partner bindings on a service:

1. Click the Partner Binding tab to associate the Partner Link Configuration resource with the partners required for the orchestration processes in the service. The Partner Binding tab is automatically populated with all partners configured in orchestration processes within the service.
2. Double click on the value in the Partner Link column to select a Partner Link Configuration resource for the partner service.

Chapter 8 Testing and Deploying Orchestration Processes

This chapter describes testing and deploying resources in a TIBCO ActiveMatrix BusinessWorks BPEL Extension project.

Topics

Overview of Testing and Deployment

TIBCO ActiveMatrix BusinessWorks provides a testing environment for stepping through your process models and determining the sources of errors. TIBCO ActiveMatrix BusinessWorks also provides a mechanism to deploy your project within your enterprise computing system by way of TIBCO Administrator.

The testing and deployment features of TIBCO ActiveMatrix BusinessWorks and TIBCO Administrator are also available for orchestration processes and service resources in the TIBCO ActiveMatrix BusinessWorks BPEL Extension. You should be familiar with TIBCO ActiveMatrix BusinessWorks and TIBCO Administrator before attempting to test and deploy your ActiveMatrix BusinessWorks BPEL Extension resources. See *TIBCO ActiveMatrix BusinessWorks Process Design Guide* and *TIBCO ActiveMatrix BusinessWorks Administration* for more information about testing and deployment.

There are some differences in the way ActiveMatrix BusinessWorks BPEL Extension resources are tested and deployed. This chapter describes the differences in testing and deployment when using the ActiveMatrix BusinessWorks BPEL Extension.

Testing Orchestration Processes

All of the features and functionality of testing process definitions described in *TIBCO ActiveMatrix BusinessWorks Process Design Guide* are available in the TIBCO ActiveMatrix BusinessWorks BPEL Extension for testing orchestration processes. For example, you can set breakpoints within orchestration processes and step through the processes as they execute. There are some additional testing features and functionality in the ActiveMatrix BusinessWorks BPEL Extension. This section describes the features available only in the ActiveMatrix BusinessWorks BPEL Extension.

Loading And Starting Orchestration Processes

Similar to process definitions with process starters in TIBCO ActiveMatrix BusinessWorks, orchestration processes require an incoming message to begin execution. Therefore, you can load orchestration processes into the test process engine, but they cannot execute until a message is received. Also, the Service resource is used to route messages to the correct orchestration process, so a Service resource must also be loaded into the test engine.

To test orchestration processes, you can create a TIBCO ActiveMatrix BusinessWorks process definition that invokes operations on a service. You can use the Invoke Service activity within a process definition to invoke a ActiveMatrix BusinessWorks BPEL Extension service. You can also use some other activity that generates a message in the proper format, such as SOAP Request Reply, if the service has endpoints exposed by way of SOAP.

When starting a test engine session, you should load the following resources in the Select Processes to Load dialog:

- a Service resource that uses one or more orchestration processes as implementations of its operations. The referenced orchestration process(es) is loaed automatically by the Service resource.
- any dynamically called TIBCO ActiveMatrix BusinessWorks sub-processes that are called by the orchestration process.
- a TIBCO ActiveMatrix BusinessWorks process definition that is configured to invoke operations on the Service resource.

Colors and Icons in Test Mode

When stepping through an orchestration process, activities and transitions in the process change color to indicate what is occurring during the execution. The colors are similar to the colors used when testing a TIBCO ActiveMatrix BusinessWorks process definition. Table 8 describes the colors and icons in test mode and their significance.

Table 8 Colors in test mode

Color/Element	Description
Black transition arrow	The transition has not yet been evaluated.
Green transition arrow	The transition has been evaluated, and its condition evaluates to true.
Red transition arrow	The transition has been evaluated, and its condition evaluates to false.
Red activity	The activity encountered an error while processing.
Bright yellow activity	The orchestration process is paused at this activity. This could be either because the activity has a breakpoint set or because the Step to Next Activity or Run To This Resource menu item was used. The activity has not yet executed, but it is the next activity to execute when execution continues.
Yellow activity	The activity is currently executing, but the focus is not on the activity. This can occur if you have multiple paths in your orchestration process and the focus is not on the current path.
	This icon indicates the activity has been skipped. This occurs when an activity's join condition evaluates to false.

Deploying TIBCO ActiveMatrix BusinessWorks BPEL Extension Resources

Deployment of ActiveMatrix BusinessWorks BPEL Extension resources is similar to deploying resources in TIBCO ActiveMatrix BusinessWorks. The following is an overview of the deployment process:

1. In TIBCO Designer, create and configure an Enterprise Archive resource, place a Process Archive into the Enterprise Archive resource, then select the Service and process resources you wish to deploy.
2. In the Enterprise Archive Configuration tab, click the Build Archive button to save the Enterprise Archive file (EAR file).
3. In TIBCO Administrator, create an application with the EAR file you created in the previous step, then deploy the application.

When creating an Enterprise Archive, loading a service or a process starter into a Process Archive will automatically load all referenced resources into the archive. For example, if you load a Service resource that uses three orchestration processes as implementations for its operations, the three orchestration processes and any resources referenced by those orchestration processes (WSDL resources, schemas, process definitions, and so on) are also loaded into the archive.

When deploying TIBCO ActiveMatrix BusinessWorks projects, please note the following:

- Service resources must be added to a Process Archive. Although it is possible to add a Service resource to a Shared Archive, the resource will not properly function when located in a Shared Archive. Add your Service resources only to Process Archives.
- When creating your Enterprise Archive file, the menu item Tools > Create Project EAR does not add Service resources to the Process Archive. You must add any Service resources to the Process Archive by selecting the Processes tab of the Process Archive and clicking the Add a Process Starter to This Archive button.
- Dynamically-called TIBCO ActiveMatrix BusinessWorks processes are not referenced directly by any resource. Therefore, these resources must be explicitly loaded into the process archive. Load dynamically-called processes by clicking the Add a Non-Process Starter to this Archive button on the Processes tab of a Process Archive resource.

See *TIBCO ActiveMatrix BusinessWorks Administration* for a complete description of how to create Enterprise Archives and how to deploy applications.

Chapter 9 Exporting and Importing Orchestration Processes

TIBCO ActiveMatrix BusinessWorks BPEL Extension can export orchestration processes and associated resources into files. The exported WS-BPEL process files comply with the WS-BPEL 2.0 schema. TIBCO ActiveMatrix BusinessWorks BPEL Extension can also import previously exported WS-BPEL files. This chapter describes how to import and export WS-BPEL resources.

Topics

Overview of Exporting and Importing

WS-BPEL is a standard, XML-based format for representing business processes. TIBCO ActiveMatrix BusinessWorks BPEL Extension can export orchestration processes and associated resources into files. The exported WS-BPEL process files comply with the WS-BPEL 2.0 schema. TIBCO ActiveMatrix BusinessWorks BPEL Extension can also import previously exported files into a project. Export and import are operations for the entire project. You cannot select specific resources to export.

While the exported WS-BPEL process files comply with the WS-BPEL 2.0 schema, TIBCO ActiveMatrix BusinessWorks BPEL Extension has implemented extensions to the WS-BPEL specification. Following are the details of the extensions:

- WS-BPEL does not define any standard way to graphically display orchestration processes. Therefore, the coordinates of a resource in the design panel are stored in an extension element in the WS-BPEL process file.
- WS-BPEL does not define any standard way to propagate context information (SOAP headers or any other business context) in the implementation process. You can propagate the context information for implementing WS-BPEL processes. For more information, refer to, Exporting Orchestration Processes, on page 111.

There is no guarantee that extensions in WS-BPEL files from other vendors will work properly when imported into TIBCO ActiveMatrix BusinessWorks BPEL Extension. Also, WS-BPEL files exported from TIBCO ActiveMatrix BusinessWorks BPEL Extension comply with the WS-BPEL 2.0 schema, but there is no guarantee that extensions in these files can be interpreted properly by products from other vendors.

Exporting Orchestration Processes

The following types of files are created during the export operation:

- WS-BPEL process files — each orchestration process in your project becomes one WS-BPEL file in the specified target export location. WS-BPEL files comply with the schema of the WS-BPEL 2.0 specification.
- WSDL files — any WSDL resources required to execute the orchestration processes are exported as WSDL files. Both abstract and concrete WSDL files are exported, whether they are used as interface descriptions for services or for bindings to external partner services. These WSDL files may also contain WS-BPEL specific extensions.
- XSD files — any schemas referenced by orchestration processes or WSDL resources are exported as XSD files. The export operation includes any schemas defined within activities of your orchestration process, such as a fault schema defined in the Throw activity (or any other referenced inline schemas). The export operation also includes input and output schemas for any invoked TIBCO ActiveMatrix BusinessWorks automated business processes.
- XSLT files — when the Input tab is used to provide mapping and transformation, the input mapping is saved as an XSLT file.

Services and Partner Link Configurations are not part of the WS-BPEL specification, and therefore these are not exported as WS-BPEL-compliant files. If you wish to export the other resources in your project for use by another TIBCO ActiveMatrix BusinessWorks project, you must use either the Project > Export Full Project or Project >Export Resources to File menu options. See *TIBCO Designer User's Guide* for more information about exporting TIBCO ActiveMatrix BusinessWorks projects.

Exporting SetContext and GetContext Activities

The GetContext and SetContext activities are exported as BPEL extension activities. The XML syntax of a BPEL extension activity is as follows:

```
<extensionActivity name =
...
...
</extensionActivity>
```

A sample XML snippet for Get Context is shown below:

```
<extensionActivity name = "GetContext">
<tibco-bpws:layout X = "187" Y = "41" iconType =
"getContextIconType"/>
<tibco-bpws:getContext>
```

```
<tibco-bpws:contextConfiguration>/context/InputHdrContext.contextR
esource</tibco-bpws:contextConfiguration>
</tibco-bpws:getContext>
</extensionActivity>
```

A sample XML snippet for Set Context is shown below:

```
<extensionActivity name = "SetContext"
 tibco-bpws:input_mapping_xslt_assoc =
"BPEL_Proc/Orch_Proc_For_Service_BPEL1-SetContext.xsl">
<tibco-bpws:layout X = "367" Y = "35" iconType =
"setContextIconType"/>
<tibco-bpws:setContext>

<tibco-bpws:contextConfiguration>/context/OutputHdrContext.context
Resource</tibco-bpws:contextConfiguration>
<tibco-bpws:showResult>false</tibco-bpws:showResult>
</tibco-bpws:setContext>
</extensionActivity>
```

To export WS-BPEL-compliant files from a project:

1. Create or choose a directory to hold the exported files. Existing files with the same name as the exported files in the specified directory are overwritten by the export process.
2. Open a TIBCO ActiveMatrix BusinessWorks project containing orchestration processes and choose **Tools > BPEL > Export** from the menu in your project.
3. In the Export BPEL dialog, choose the directory in step 1 to store the exported files. If the directory does not exist (because you did not pre-create the directory in step 1 or you chose a different directory), an error is returned stating that the directory path does not exist. If the directory is not empty, a warning is given asking if you wish to proceed.
4. Upon completion of the export, navigate to the directory and view the exported files.

Importing Orchestration Processes

TIBCO ActiveMatrix BusinessWorks BPEL Extension can import WS-BPEL files that were previously exported from TIBCO ActiveMatrix BusinessWorks BPEL Extension. Each vendor that implements WS-BPEL has a variety of extensions. Therefore, TIBCO ActiveMatrix BusinessWorks BPEL Extension cannot guarantee that WS-BPEL projects from other vendors can be successfully imported.

To import a WS-BPEL project:

1. Open a TIBCO ActiveMatrix BusinessWorks project and choose **Tools > BPEL > Import** from the menu. The Import BPEL dialog appears. This dialog has two fields, Import From BPEL Project and Import to New Repository.
2. Use the **Browse** button in the Import From BPEL Project field to select the location of a previously exported WS-BPEL project. This field specifies the source files to import.
3. Use the **Browse** button to select the TIBCO ActiveMatrix BusinessWorks project repository as the target of the import. You can choose the currently opened project as the target location, if desired. If there are any resources in the target project that have the same name as resources in the imported project, a warning is given that the resources will be overwritten.
4. Once the import operation is complete, the target project will contain the imported orchestration processes and associated resources.

Predefined process variables, such as `$_processContext` can not be represented in exported files. Therefore, these items are replaced by `$VARIABLE_NOT_FOUND`. When exported files contain `$VARIABLE_NOT_FOUND`, you must fix the expressions that use these variables when the files are imported into a project.

Chapter 10 Palette Reference

This chapter describes the resources in the ActiveMatrix BusinessWorks BPEL Extension palette. This chapter also describes the Service resource and the Partner Link Configuration resource in the Service palette. The Service resource has a different interface when the ActiveMatrix BusinessWorks BPEL Extension is used.

Topics

Assign

Activity

The Assign activity can copy or modify data from one process variable to another process variable. To perform an assignment:

1. Drag and drop an Assign activity into an orchestration process.
2. Click the + button to add an assignment to the table in the Assignments field. Use the X button to delete assignments, and use the arrow buttons to move assignments in the list.
3. In the right-hand corner of the Assign From field, click the XPath Formula Builder to create an XPath expression. This expression obtains data from a process variable.
4. In the right-hand corner of the Assign To field, click the XPath Formula Builder to create an XPath expression. This expression sets the process variable to the desired value.
5. Perform step 2 through step 4 to create or delete more assignments within the Assign activity.

See Assigning a Value to a Variable on page 56 for more information about the Assign activity and process variables.

Configuration

The Configuration tab has the following fields.

Field	Description
Name	The name to appear as the label for the activity.
Description	Short description of the activity.
Assignments	Use the + button to add assignments to this activity. Use the X button to delete the selected assignment, and use the Arrow buttons to move assignments in the list. This field contains a table that specifies assignments from process variables to other process variables. Use the XPath Formula Builder button in the right-hand corner of each field to create expressions for each assignment.

Join Condition

The Join Condition tab allows you to define an expression based on the values of the incoming transitions to the activity. If the expression evaluates to true, the activity is executed, if the expression evaluates to false, either the activity is bypassed or a fault is thrown. See Transitions and Join Conditions on page 60 for more information about join conditions.

The Join Condition tab has the following fields.

Field	Description
Join Action	Specifies what action to perform if the join expression evaluates to false. The following options are available: • Inherit setting — use the default action specified within the current scope. The current scope is either the current group containing the activity or the orchestration process if the activity is not within a group. • Bypass activity if false — skip the processing of the activity, set the outgoing transition conditions to false without evaluating them, and resume processing. • Throw fault if false — throw a joinFailure fault and switch control to an error-handling routine configured to catch all unhandled exceptions. If the exception is not caught, the fault is propagated to the calling environment.
Join Expression	The expression to evaluate to determine if the activity should be executed. The only data available to this expression is the value of any transitions into the activity. You can use the XPath Formula Builder to create the join expression.

Catch

Activity

The Catch activity receives control of execution when an exception occurs. You can select a specific exception type to catch or you can specify that this activity should catch all unhandled exceptions. You can have more than one Catch activity in each exception scope, but each Catch activity must have a unique exception type.

The Catch activity allows you to transition to activities you wish to perform to handle the exception. Transitions are not permitted between Catch error-handling routines within a scope, and you can not transition back to the main execution path from the Catch error-handling routine.

If you wish to propagate the caught exception to the next highest scope, use the Rethrow activity.

See Chapter 6, Exception Handling, on page 85 for more information about error handling and using the Catch activity.

Configuration

The Configuration tab has the following fields.

Field	Description
Name	The name to appear as the label for the activity.
Description	Short description of the activity.
Catch All	Checking this box indicates you wish processing to resume with this activity when any exception is encountered that is not already handled by another Catch activity. You can have only one Catch activity within an exception scope that has this field checked.
Fault Name	Specifies the exception type to catch. The list of available exceptions that can be raised in the current scope is automatically placed in the drop-down list in this field. This field is not available when the Catch All field is checked. Each Catch activity within an exception scope must specify a different value for this field.

Output

The output for the activity is the following.

Output Item	Datatype	Description
<exceptionName>	varies	Contains the schema for the thrown exception. The contents of this element vary depending on the exception that is encountered.

Checkpoint

Activity

The Checkpoint activity performs a checkpoint in a running process instance. A checkpoint saves the current process data and state so that it can be recovered at a later time in the event of a failure. If a process engine fails, all process instances can be recovered and resume execution at the location of their last checkpoint in the orchestration process. If a process instance fails due to an unhandled exception or manual termination, it can optionally be recovered at a later time, if the process engine is configured to save checkpoint data for failed processes. See *TIBCO ActiveMatrix BusinessWorks Administration* for more information about recovering failed process instances.

Only the most recent state is saved by a checkpoint. If you have multiple checkpoints in a process, only the state from the last checkpoint is available for recovering the process.

Placing a Checkpoint activity between a Receive activity and a Reply activity will result in a failure in the reply message if the process is recovered using the checkpoint. That is, a reply message must be sent from the same process that received the message, and a restart of the process causes an error. If your checkpoint must remain between the receive and reply, consider changing the WSDL to two one-way operations and use an Invoke activity to send the reply message.

Configuration

The Configuration tab has the following fields.

Field	Description
Name	The name to appear as the label for the activity.
Description	Short description of the activity.

Join Condition

The Join Condition tab allows you to define an expression based on the values of the incoming transitions to the activity. If the expression evaluates to true, the activity is executed, if the expression evaluates to false, either the activity is bypassed or a fault is thrown. See Transitions and Join Conditions on page 60 for more information about join conditions.

The Join Condition tab has the following fields.

Field	Description
Join Action	Specifies what action to perform if the join expression evaluates to false. The following options are available: • Inherit setting — use the default action specified within the current scope. The current scope is either the current group containing the activity or the orchestration process if the activity is not within a group. • Bypass activity if false — skip the processing of the activity, set the outgoing transition conditions to false without evaluating them, and resume processing. • Throw fault if false — throw a joinFailure fault and switch control to an error-handling routine configured to catch all unhandled exceptions. If the exception is not caught, the fault is propagated to the calling environment.
Join Expression	The expression to evaluate to determine if the activity should be executed. The only data available to this expression is the value of any transitions into the activity. You can use the XPath Formula Builder to create the join expression.

Input

The input for the activity is the following:

Input Item	Datatype	Description
duplicateKey	string	A key value that is used to compare to other process instances to determine whether another process instance with the same duplicateKey value already exists. See *TIBCO ActiveMatrix BusinessWorks Process Design Guide* for more information on detecting duplicate process instances.

Error Output

The Error Output tab lists the possible exceptions that can be thrown by this activity.

Exception	Description
DuplicateException	This exception is thrown in the event that the process instance is restarted and another process instance with the same value for the duplicateKey has been detected. The following is the schema of this exception: • `msg` — an error message indicating a duplicate process instance has been detected. • `msgCode` — code for the error message. • `duplicateKey` — value of the duplicateKey for the duplicate process. • `previousJobID` — ID of the process that already exists that has the same duplicateKey.

Exit

Activity

The Exit activity terminates the orchestration process immediately. All currently executing activities are halted.

Configuration

The Configuration tab has the following fields.

Field	Description
Name	The name to appear as the label for the activity.
Description	Short description of the activity.

Join Condition

The Join Condition tab allows you to define an expression based on the values of the incoming transitions to the activity. If the expression evaluates to true, the activity is executed, if the expression evaluates to false, either the activity is bypassed or a fault is thrown. See Transitions and Join Conditions on page 60 for more information about join conditions.

The Join Condition tab has the following fields.

Field	Description
Join Action	Specifies what action to perform if the join expression evaluates to false. The following options are available: • Inherit setting — use the default action specified within the current scope. The current scope is either the current group containing the activity or the orchestration process if the activity is not within a group. • Bypass activity if false — skip the processing of the activity, set the outgoing transition conditions to false without evaluating them, and resume processing. • Throw fault if false — throw a joinFailure fault and switch control to an error-handling routine configured to catch all unhandled exceptions. If the exception is not caught, the fault is propagated to the calling environment.
Join Expression	The expression to evaluate to determine if the activity should be executed. The only data available to this expression is the value of any transitions into the activity. You can use the XPath Formula Builder to create the join expression.

Invoke Process

Activity

The Invoke Process activity invokes a TIBCO ActiveMatrix BusinessWorks automated process. This activity spawns a new process instance to execute the specified process definition. You must pass any desired data into the process instance by way of an input variable or input mapping. Process variables from the current orchestration process are not available to the called TIBCO ActiveMatrix BusinessWorks process.

Configuration

The Configuration tab has the following fields.

Field	Description
Name	The name to appear as the label for the activity.
Description	Short description of the activity.
Process Name	The process definition you would like to call. You can browse the available process definitions.
Process Name Dynamic Override	An XPath formula specifying the name of the process to call. Use this field to dynamically determine which process to call when the process instance is running. See Dynamically Determining the Process to Call on page 127 for more information about this field.
Wait For Completion	Checking this box indicates that the called process definition must complete before proceeding to the next activity in the orchestration process. This option is only available for process definitions that do not produce output. That is, for process definitions that have a schema defined on the Input Editor tab of the End activity, you must wait for their completion before continuing. For process definitions that have no output, you can optionally uncheck this field to indicate that the process definition can execute asynchronously and the orchestration process can continue on to the next activity.

Field	Description
Custom Id	This field can contain an XPath expression that specifies a custom ID for the process instance. This ID is displayed in the View Service dialog of TIBCO Administrator, and it is also available in the `$_processContext` process variable in the spawned process instance.
Invoke Message Correlations	Correlations for the outgoing message to the process definition. See Correlations on page 62 for more information about correlations.
Reply Message Correlations	Correlations for the incoming message received from the process definition. This field is only available when the process definition selected contains an output schema. See Correlations on page 62 for more information about correlations.
Custom Icon File	File to use for the Invoke Process activity in the orchestration process. This allows you to customize the look of your activity with your own images.

Dynamically Determining the Process to Call

You can use the Process Name Dynamic Override field on the Configuration tab to specify an XPath expression that determines which process to call. This is useful for calling a different process depending upon the value of the XPath expression. For example, for incoming orders over $10,000, you wish to call a process that includes additional approval tasks. For orders under $10,000, you wish to call a process that handles the order automatically. If the Receive Starter is named IncomingOrder and the order amount is stored in a field named orderAmount, then your XPath expression to determine which process to call may look like the following:

```
if($IncomingOrder/orderAmount > 10000) then
            '/MyProject/manualApproval.process'
      else '/MyProject/processOrder.process'
```

Use the full path and name of the process file as stored in the project directory.

When you use the Process Name Dynamic Override field, you must also specify a process to call in the Process field. The input, output, and error definitions of the specified process must be the same as any process that the expression in the Process Name Dynamic Override field can evaluate to.

It may be helpful to create process definitions that act as programmatic interfaces when using the Process Name Dynamic Override field.

In the example above, you may create a process named orderProcessOrApprove. The only purpose of this process is for specifying the input, output, and error schemas. Place this process in the Process field of the Call Process tab. Then, create the manualApproval and processOrder process definitions as copies of the orderProcessOrApprove process. The Call Process activity then has the correct input, output, and error schemas for all processes that can be called.

If you use the Process Name Dynamic Override field, make sure you include all potentially callable subprocesses when you create your Process Archive for deployment. TIBCO Designer cannot determine which subprocesses are potentially callable at design time, and therefore they cannot be automatically included in a process archive. See *TIBCO Designer User's Guide* for more information about creating process archives.

Join Condition

The Join Condition tab allows you to define an expression based on the values of the incoming transitions to the activity. If the expression evaluates to true, the activity is executed, if the expression evaluates to false, either the activity is bypassed or a fault is thrown. See Transitions and Join Conditions on page 60 for more information about join conditions.

The Join Condition tab has the following fields.

Field	Description
Join Action	Specifies what action to perform if the join expression evaluates to false. The following options are available: • Inherit setting — use the default action specified within the current scope. The current scope is either the current group containing the activity or the orchestration process if the activity is not within a group. • Bypass activity if false — skip the processing of the activity, set the outgoing transition conditions to false without evaluating them, and resume processing. • Throw fault if false — throw a joinFailure fault and switch control to an error-handling routine configured to catch all unhandled exceptions. If the exception is not caught, the fault is propagated to the calling environment.
Join Expression	The expression to evaluate to determine if the activity should be executed. The only data available to this expression is the value of any transitions into the activity. You can use the XPath Formula Builder to create the join expression.

Input Variable

The Input Variable tab allows you to specify whether you will use the contents of a process variable as input to this activity or you will use the Input tab to specify input bindings as XPath expressions. If you choose to use a process variable, the schema of the variable must match the input schema of the activity exactly.

Field	Description
Input Style	This field allows you to select one of the following input types: • Use Input Variable — specifies that you will select an input variable to use as input for the activity. The schema of the variable must match the input schema of the activity or an error is thrown when this activity is executed. • Use Input Binding Tab — specifies that you will use the Input tab to create mappings and XPath expressions to provide input for this activity.
Input Variable	When Use Input Variable is selected in the Input Style field, this field allows you to select from a list of process variables to use as the input for the activity.

Input

The Input tab is only enabled if Use Input Binding Tab is selected in the Input Style field on the Input Variable tab. See *TIBCO ActiveMatrix BusinessWorks Process Design Guide* for more information about mapping and transforming input data.

The input for the activity is the following.

Input Item	Datatype	Description
input	varies	The input to the called process is defined by the Output Editor tab in the called process' Start activity. See *TIBCO ActiveMatrix BusinessWorks Process Design Guide* for more information.

Output

The output for the activity is the following.

Output Item	Datatype	Description
output	varies	The output of the called process is defined by the Input Editor tab in the called process' End activity. See *TIBCO ActiveMatrix BusinessWorks Process Design Guide* for more information.

Output Variable

The output variable tab allows you to select a process variable to contain the output of the activity. The schema of the process variable must match the schema of the output for the activity shown on the Output tab.

Invoke Service

Activity

The Invoke Service activity is used within a TIBCO ActiveMatrix BusinessWorks process definition to invoke a ActiveMatrix BusinessWorks BPEL Extension service.

Configuration

The Configuration tab has the following fields.

Field	Description
Name	The name to appear as the label for the activity.
Description	Short description of the activity.
Service	Service resource that you wish to invoke. Use the Browse button to select from a list of available Service resources in the project. In the Select a Resource dialog, you can select the desired Port and Operation to invoke. This automatically populates the Endpoint and Operation fields in the Configuration tab.
Endpoint	The Endpoint of the service you wish to use. This is automatically populated based on the Port you select in the Select a Resource dialog. You can change the Endpoint to a different port by selecting an available endpoint from the drop-down list.
Operation	The operation you wish to invoke on the selected service. This is automatically populated based on the operation you selected in the Select a Resource dialog. You can change the operation by selecting another available operation from the drop-down list.
Timeout (sec)	Time to wait (in seconds) for the operation to complete.

Input

See *TIBCO ActiveMatrix BusinessWorks Process Design Guide* for more information about mapping and transforming input data.

The input for the activity is the following.

Input Item	Datatype	Description
input message	complex	The input message of the operation. This element contains all input message parts for the operation.

Output

The output for the activity is the following.

Output Item	Datatype	Description
output message	complex	The output message of the operation. This element contains all output message parts for the operation.

Error Output

The Error Output tab lists the possible exceptions that can be thrown by this activity.

Exception	Description
MessageFault	This exception is thrown when the invoked service replies with a fault. The fault message is returned in the `MessageFault/fault/Error/Message` element.

Invoke

Activity

The Invoke activity invokes a partner service. The partner service can be local (a service within the same project), it can be a third-party service invoked by way of SOAP over the Internet, or it can be an internal service invoked by way of SOAP over the corporate intranet.

You can specify a binding to a concrete WSDL to invoke a specific partner service, or you can use an abstract partner portType and use a Partner Link Configuration resource to bind the invocation request to the correct partner service within the Service resource. See Partners on page 96 for more information about using partner services.

You can also enable WS-Security for the Invoke activity by associating a security policy with a service endpoint operation defined for a Partner Link in the Partner Link Configuration resource. When a security policy is associated, the WS-Security processing is enabled and performed according to WS-Security guidelines for the outbound, inbound, and inbound fault message exchange specified in the Security Policy Association resource.

Security policies in TIBCO ActiveMatrix BusinessWorks BPEL Extension are associated in exactly the same way as in TIBCO ActiveMatrix BusinessWorks. For more information on how to associate a security policy, refer to the TIBCO ActiveMatrix BusinessWorks documentation.

Configuration

The Configuration tab has the following fields.

Field	Description
Name	The name to appear as the label for the activity.
Description	Short description of the activity.
Partner	The partner declared on the Orchestration Process that you wish to invoke. See Partners on page 96 for more information about declaring partners.
Operation	The operation on the specified partner that you wish to invoke.
Invoke Message Correlations	Correlations for the outgoing message to the partner service. See Correlations on page 62 for more information about correlations.

Field	Description
Reply Message Correlations	Correlations for the reply message received from the partner service. This field is only available when the operation selected contains a reply message. See Correlations on page 62 for more information about correlations.

Join Condition

The Join Condition tab allows you to define an expression based on the values of the incoming transitions to the activity. If the expression evaluates to true, the activity is executed, if the expression evaluates to false, either the activity is bypassed or a fault is thrown. See Transitions and Join Conditions on page 60 for more information about join conditions.

The Join Condition tab has the following fields.

Field	Description
Join Action	Specifies what action to perform if the join expression evaluates to false. The following options are available: • Inherit setting — use the default action specified within the current scope. The current scope is either the current group containing the activity or the orchestration process if the activity is not within a group. • Bypass activity if false — skip the processing of the activity, set the outgoing transition conditions to false without evaluating them, and resume processing. • Throw fault if false — throw a joinFailure fault and switch control to an error-handling routine configured to catch all unhandled exceptions. If the exception is not caught, the fault is propagated to the calling environment.
Join Expression	The expression to evaluate to determine if the activity should be executed. The only data available to this expression is the value of any transitions into the activity. You can use the XPath Formula Builder to create the join expression.

Input Variable

The Input Variable tab allows you to specify whether you will use the contents of a process variable as input to this activity or you will use the Input tab to specific input bindings as XPath expressions. If you choose to use a process variable, the schema of the variable must match the input schema of the activity exactly.

Field	Description
Input Style	This field allows you to select one of the following input types: • Use Input Variable — specifies that you will select an input variable to use as input for the activity. The schema of the variable must match the input schema of the activity or an error is thrown when this activity is executed. • Use Input Binding Tab — specifies that you will use the Input tab to create mappings and XPath expressions to provide input for this activity.
Input Variable	When Use Input Variable is selected in the Input Style field, this field allows you to select from a list of process variables to use as the input for the activity.

Input

The Input tab is only enabled if Use Input Binding Tab is selected in the Input Style field on the Input Variable tab. See *TIBCO ActiveMatrix BusinessWorks Process Design Guide* for more information about mapping and transforming input data.

The input for the activity is the following.

Input Item	Datatype	Description
input	varies	The input message for the desired operation on the partner service. The schema depends upon the message defined in the WSDL for the partner service.

Output

The output for the activity is the following.

Output Item	Datatype	Description
output	varies	The output message for the desired operation on the partner service. The schema depends upon the message defined in the WSDL for the partner service.

Output Variable

The output variable tab allows you to select a process variable to contain the output of the activity. The schema of the process variable must match the schema of the output for the activity shown on the Output tab.

Error Output

The Error Output tab lists the possible exceptions that can be thrown by this activity.

Exception	Description
faultName	This exception is thrown when the invoked service replies with a fault. The fault message is returned in the *faultName*`/MessageFault/fault/Error/Message` element.

Null

Activity

The Null activity is an activity with no action performed. This activity has a name and a description specified on the Configuration tab, but there is no input or output for the activity.

This activity can be useful in error-handling routines if you wish to catch a fault but perform no action. This activity can also be useful to join multiple process flows.

Configuration

The Configuration tab has the following fields.

Field	Description
Name	The name to appear as the label for the activity.
Description	Short description of the activity.

Join Condition

The Join Condition tab allows you to define an expression based on the values of the incoming transitions to the activity. If the expression evaluates to true, the activity is executed, if the expression evaluates to false, either the activity is bypassed or a fault is thrown. See Transitions and Join Conditions on page 60 for more information about join conditions.

The Join Condition tab has the following fields.

Field	Description
Join Action	Specifies what action to perform if the join expression evaluates to false. The following options are available: • Inherit setting — use the default action specified within the current scope. The current scope is either the current group containing the activity or the orchestration process if the activity is not within a group. • Bypass activity if false — skip the processing of the activity, set the outgoing transition conditions to false without evaluating them, and resume processing. • Throw fault if false — throw a joinFailure fault and switch control to an error-handling routine configured to catch all unhandled exceptions. If the exception is not caught, the fault is propagated to the calling environment.
Join Expression	The expression to evaluate to determine if the activity should be executed. The only data available to this expression is the value of any transitions into the activity. You can use the XPath Formula Builder to create the join expression.

On Alarm

Activity

The On Alarm activity activates when the specified time is reached. This activity can be used to set an alarm for a specific time or to periodically perform a set of actions. This activity can be useful for marking time periods for reminders or to set a timeout for executing an orchestration process.

The On Alarm activity starts an event-handling routine. You can transition to one or more activities to perform after the alarm occurs, but you cannot transition from activities within the event handling routine to the main processing flow.

The On Alarm activity is active only when the scope where it is located is executing. For example, if the On Alarm activity appears within a group, the alarm will go off only as long as the group continues to execute. The alarm is not checked until the group starts to execute, and once processing in the group is complete, the alarm is no longer active.

Configuration

The Configuration tab has the following fields.

Field	Description
Name	The name to appear as the label for the activity.
Description	Short description of the activity.

Field	Description
Style	Specifies the style of determining when to end the alarm period. Can be one of the following: • For — specifies an amount of time in years, days, months, hours, minutes, seconds, and milliseconds that this alarm should remain active. • Until — specifies a date until which this alarm should remain active. When For is specified, the alarm goes off at the end of the amount of time specified. For example, if you specify 1 day, the alarm goes off one day after the current scope starts to execute. When Until is specified, the alarm goes off on the specific day and time you provide. For example, if all processes must provide a status at the end of the day, you would specify an expression that evaluates to the desired time on the current date.
For Config Option	You can specify one of the following: • Literal — this allows you to provide literal values, such as 1 day or 3 months. • Expression — this allows you to provide an XPath expression that evaluates to an amount of time.
Until Config Option	You can specify one of the following: • Literal — this allows you to choose a specific date and time from a calendar and clock. • Expression — this allows you to provide an XPath expression that evaluates to a specific date and time.
For	Specify the amount of time in the provided fields: Years, Months, Days, Hours, Minutes, Seconds, Milliseconds.
For Expression	Specify an XPath expression that evaluates to an amount of time. Use the XPath Formula Builder button to bring up the formula editor, if desired.

Field	Description
Until	Specify a specific date using the calendar and clock provided.
Until Expression	Specify an XPath expression that evaluates to a specific date and time. Use the XPath Formula Builder to bring up the formula editor, if desired.
Repeat	Check this box if you wish the alarm to repeat at periodic intervals after the initial alarm goes off.
RepeatEvery ConfigOption	You can specify one of the following: • Literal — this allows you to choose an amount of time in years, months, days, hours, minutes, seconds, and milliseconds. • Expression — this allows you to provide an XPath expression that evaluates to an amount of time.
RepeatEvery Expression	Specify an XPath expression that evaluates to an amount of time. Use the XPath Formula Builder button to bring up the formula editor, if desired. The alarm will repeat after the specified amount of time is reached.
RepeatEvery	Specify the amount of time in the provided fields: Years, Months, Days, Hours, Minutes, Seconds, Milliseconds.

On Event

Activity

The On Event activity activates when a message is received from the specified input partner. This activity is used for potential messages that the process does not require. For example, a requestor may send a message that asks for the current status of an orchestration process.

The On Event activity starts an event-handling routine. You can transition to one or more activities to perform after the event occurs, but you cannot transition from activities within the event handling routine to the main processing flow.

The On Event activity is active only when the scope where it is located is executing. For example, if the On Event activity appears within a group, the events will be received only as long as the group continues to execute. The events are not received until the group starts to execute, and once processing in the group is complete, the events are no longer received.

Configuration

The Configuration tab has the following fields.

Field	Description
Name	The name to appear as the label for the activity.
Description	Short description of the activity.
Input partner	The input partner declared on the Orchestration Process that describes the incoming message. See Partners on page 96 for more information about declaring input partners.
Operation	The operation on the specified input partner that was invoked to produce the incoming message.
Message Exchange	A string used to associate reply messages with the correct activity that received the message. Any replies to the incoming message should use the same string in the Message Exchange field to ensure that the reply is sent to the correct recipient.
Correlations	Correlations for the incoming message. See Correlations on page 62 for more information about correlations.

Output

The output for the activity is the following.

Output Item	Datatype	Description
output	varies	The output of this activity is the incoming message from the calling client or service. The structure of the message is determined by the schema of the input message for the selected operation on the Configuration tab.

Output Variable

The output variable tab allows you to select a process variable to contain the output of the activity. The schema of the process variable must match the schema of the output for the activity shown on the Output tab.

Orchestration Process

Resource

Orchestration processes describe communication between services and operations to perform a given business process. An orchestration process is a flow of work that accomplishes some task.

Orchestration processes are executed when an incoming message is received. Therefore, an orchestration process typically begins with one Receive Starter activity, but there could be more than one Receive starter that starts the process.

Configuration

The Configuration tab has the following fields.

Field	Description
Name	The name to appear as the label for the resource.
Description	Short description of the resource.
Custom Icon File	File to use for the icon of the Orchestration Process. This allows you to customize the look of your processes with your own images.
Namespace Registry	Clicking the Edit button in this field allows you to view, add, change or delete the namespaces used in the input partners defined for this orchestration process. You can also view and edit schema and WSDL imports.
Target Namespace	Target namespace of this orchestration process. This namespace is used when exporting the process to a WS-BPEL file.

Field	Description
Default Join Action	Specifies the default join condition for activities in the orchestration process. See Transitions and Join Conditions on page 60 for more information about join conditions. Choose one of the following: • Bypass activity if false — skip the processing of the activity, set the outgoing transition conditions to false without evaluating them, and resume processing. • Throw fault if false — throw a joinFailure fault and switch control to an error-handling routine configured to catch all unhandled exceptions. If the exception is not caught, the fault is propagated to the calling environment.

Process Variables

The Process Variables tab allows you to define variables to hold data while executing the orchestration process. Process variables can be used to maintain process state, pass input or hold output from activities, or perform other functions within the orchestration process.

Process variables are defined in the same way as TIBCO ActiveMatrix BusinessWorks process variables. See Process Variables on page 55 for more information about defining and using process variables.

Input Partners

The Input Partners tab allows you to specify the PortTypes that this process implements. These PortTypes define the types of input and output messages that can be processed. See Partners on page 96 for more information about input partners.

The Input Partners tab has the following fields.

Field	Description
Input Partners	Use the + button to add input partners, use the X button to remove input partners, and use the arrow buttons to move input partners in the list. Once an input partner has been added, double click the Input Partner Name field to specify a name for the input partner. Use the PortType field to associate a PortType with the selected Input Partner.
PortType	This field associates a PortType with an input partner selected in the Input Partners field. Use the Browse button to select a WSDL resource that describes the appropriate PortType. The abstract portion of the WSDL resource is used, and any concrete bindings are ignored, if they exist in the WSDL. Select the desired PortType from the list of PortTypes in the Select a Resource dialog. You can later change the PortType by selecting from a list of available PortTypes in the drop-down list in the PortType field.

Partners

The Partners tab is used to declare any partner services that will be invoked by activities within the orchestration process. You can bind the specified partners to specific ports, or you can use Partner Link Configuration resources. See Partners on page 96 for more information about partners.

The Partners tab has the following fields.

Field	Description
Partners	Use the + button to add partners, use the X button to remove partners, and use the arrow buttons to move partners in the list. Double click on the value in the Partner Name column to change the name of a partner. Select a PortType for the partner in the PortType field. If you wish to specify a concrete binding to a specific endpoint for this partner, double click on the Partner Link field and use the Browse button to locate the endpoint.
PortType	Use the Browse button to select the WSDL for the partner. Select the appropriate PortType in the Select a Resource field. If you wish to change the PortType, you can select a valid PortType from the drop-down list in the PortType field.

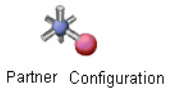
Correlations

The Correlations tab is used to declare correlations for the orchestration process. These correlations can then be later set by activities that receive or send messages. Correlations are used to route messages to the appropriate orchestration process. See Correlations on page 62 for more information about correlations.

Use the + button to add correlations, use the X button to delete correlations, and use the arrow buttons to move correlations in the list. Double click on the Correlation Name column to change the name of a correlation to the desired value.

Partner Link Configuration

Resource



The Partner Link Configuration associates abstract partner portTypes with concrete port bindings. This allows you to easily link to new partner services without changing orchestration processes that invoke the partners.

See Partners on page 96 for more information about partners and Partner Link Configuration resources.

Configuration

The Configuration tab has the following fields.

Field	Description
Name	The name to appear as the label for the resource.
Description	Short description of the resource.
Partner Links	Use the + button to add partner links to the list, use the X button to delete partner links from the list, use the arrow buttons to move partner links in the list. Specify the following in the columns of the table for each partner: • Name — Double click on this field to specify a name for the partner. • Timeout (seconds) — Double click on this field to specify a timeout in seconds for invocations of this partner. By default, the timeout is zero (0) which provides an unlimited amount of time for invocations. • Service Endpoint — The endpoint of the service. You can use concrete WSDL resources with port bindings or you can specify local service resources with Local specified in the endpoint binding.
Additional Configuration for HTTP Endpoints	
Use HTTP Proxy	Specifies to use a proxy server to gain access outside of a firewall.

Field	Description
Proxy Name	Host name or IP address of the proxy server.
Use Basic Authentication	Specifies that the invocation of the service must provide authentication to the service.
Identity	When basic authentication (username and password) is required, provide an Identity resource in this field that contains the appropriate username and password. See *TIBCO Designer Palette Reference* for more information.
SSL	When SSL is required for authentication to the service, click the Configure SSL button to supply the appropriate properties. See HTTP SSL Configuration Fields on page 150 information more information.
HTTP SSL Configuration Fields	
Trusted Certificates Folder	Folder in the project containing one or more certificates from trusted certificate authorities. This folder is checked when an invocation is made to the partner service to ensure that the server is trusted. This prevents connections to rogue servers.
Identity	This is an identity resource that contains the clients's digital certificate and private key. See *TIBCO Designer Palette Reference* for more information.
Strong Cipher Suites Only	Specifies that only cipher suites with strong encryption should be used, if they are available on the host you are connecting to.
Additional Configuration for JMS Endpoints	
JNDI Tab Context URL	URL for the JNDI connection to the JMS server (`javax.naming.Context.PROVIDER_URL`).
JNDI Tab User Name	User name for logging into the JNDI server (`javax.naming.Context.SECURITY_PRINCIPAL`). If the JNDI provider does not require access control, this field can be empty.

Field	Description
JNDI Tab Password	Password for logging into the JNDI server (`javax.naming.Context.SECURITY_CREDENTIALS`). If the JNDI provider does not require access control, this field can be empty.
JNDI Tab SSL	When SSL is required for authentication to the service, click the Configure SSL button to supply the appropriate SSL JMS SSL Configuration Fields on page 152 information. See for more information.
JNDI Tab Validate Security Context	Some application servers store the security context on the thread used to establish the JNDI connection (at the time of this release, only the WebLogic application server does this). In that case, the first activity to use this resource establishes the security context, then subsequent activities use the same security context, unless this field is checked. Checking this field ensures that each activity that uses this resource examines the security context to determine if the activity uses the same security context as the security context established on the thread. If they are different, the activity's configured security context is used. Checking this field causes additional overhead for activities that use this resource. Therefore, only check this field when necessary.
JMS Tab User Name	User name to use when logging into the JMS server. If the JMS provider does not require access control, this field can be empty.
JMS Tab Password	Password to use when logging into the JMS server. If the JMS provider does not require access control, this field can be empty.

Field	Description
JMS Tab Delivery Mode	The delivery mode of the message. Can be one of the following: • Persistent — signifies the messages are stored and forwarded. • Non-Persistent — messages are not stored and may be lost due to failures in transmission. • TIBCO EMS Reliable— this mode is only available when using TIBCO Enterprise Message Service. See the TIBCO Enterprise Message Service documentation for more information about this mode.
JMS Tab Expiration	Corresponds to JMSExpiration property that specifies how long the message can remain active (in seconds). If set to 0, the message does not expire. This field is set in seconds, but the JMSExpiration property is displayed in milliseconds.
JMS Tab Priority	Priority of the message. You may set the priority to a value from 0-9. The default value is 4.
JMS SSL Configuration Fields	
Basic Tab Trusted Certificates Folder	Folder in the project containing one or more certificates from trusted certificate authorities. This folder is checked when an invocation is made to the partner service to ensure that the server is trusted. This prevents connections to rogue servers.
Basic Tab Identity	This is an identity resource that contains the clients's digital certificate and private key. See *TIBCO Designer Palette Reference* for more information.
Advanced Tab Trace	Specifies whether SSL tracing should be enabled during the connection. If checked, the SSL connection messages are logged and sent to the console.

Field	Description
Advanced Tab Debug Trace	Specifies whether SSL debug tracing should be enabled during the connection. Debug tracing provides more detailed messages than standard tracing.
Advanced Tab Verify Host Name	This field specifies to check the host name of the server against the host name listed in the server's digital certificate. This provides additional verification that the host name you believe you are connecting to is in fact the desired host. If the host name specified in the endpoint binding is not an exact match to the host name specified in the server's digital certificate, the connection is refused. Note: If an equivalent host name (for example, an IP address) is specified in the endpoint binding, but the name is not an exact match of the hostname in the host's digital certificate, the connection is refused.
Advanced Tab Expected Host Name	Specifies the name of the host you are expecting to connect to. This field is only relevant if the Verify Host Name field is also checked. If the name of the host in the host's digital certificate does not match the value specified in this field, the connection is refused. This prevents hosts from attempting to impersonate the host you are expecting to connect to.
Advanced Tab Strong Cipher Suites Only	When checked, this field specifies that the minimum strength of the cipher suites used can be specified with the `bw.plugin.security.strongcipher.minstrength` custom engine property. See *TIBCO ActiveMatrix BusinessWorks Administration* for more information about this property. The default value of the property disables cipher suites with an effective key length below 128 bits. When this field is unchecked, only cipher suites with an effective key length of up to 128 bits can be used.

Receive Starter

Activity

The Receive Starter activity begins an orchestration process when a message is received. Typically you have one Receive starter activity for each orchestration process, but you could have more than one to indicate that multiple messages are required for the orchestration process and any of them can start the execution of the process.

Configuration

The Configuration tab has the following fields.

Field	Description
Name	The name to appear as the label for the activity.
Description	Short description of the activity.
Input Partner	The input partner declared on the Orchestration Process that describes the incoming message. See Partners on page 96 for more information about declaring input partners.
Operation	The operation on the specified input partner that was invoked to produce the incoming message.
Message Exchange	A string used to associate reply messages with the correct activity that received the message. Any replies to the incoming message should use the same string in the Message Exchange field to ensure that the reply is sent to the correct recipient.
Correlations	Correlations for the incoming message. See Correlations on page 62 for more information about correlations.

Output

The output for the activity is the following.

Output Item	Datatype	Description
output	varies	The output of this activity is the incoming message from the calling client or service. The structure of the message is determined by the schema of the input message for the selected operation on the Configuration tab.

Output Variable

The output variable tab allows you to select a process variable to contain the output of the activity. The schema of the process variable must match the schema of the output for the activity shown on the Output tab.

Receive

Activity

The Receive activity waits for a message. The orchestration process does not continue to the next activity until the message is received. For example, your orchestration process may handle new requests for loans. A message requesting a new loan starts the process, but later in the process, an approval is required from a loan officer. You would use the Receive activity to wait for the approval message.

Configuration

The Configuration tab has the following fields.

Field	Description
Name	The name to appear as the label for the activity.
Description	Short description of the activity.
Input Partner	The input partner declared on the Orchestration Process that describes the incoming message. See Partners on page 96 for more information about declaring input partners.
Operation	The operation on the specified input partner that was invoked to produce the incoming message.
Message Exchange	A string used to associate reply messages with the correct activity that received the message. Any replies to the incoming message should use the same string in the Message Exchange field to ensure that the reply is sent to the correct recipient.
Correlations	Correlations for the incoming message. See Correlations on page 62 for more information about correlations.

Join Condition

The Join Condition tab allows you to define an expression based on the values of the incoming transitions to the activity. If the expression evaluates to true, the activity is executed, if the expression evaluates to false, either the activity is bypassed or a fault is thrown. See Transitions and Join Conditions on page 60 for more information about join conditions.

The Join Condition tab has the following fields.

Field	Description
Join Action	Specifies what action to perform if the join expression evaluates to false. The following options are available: • Inherit setting — use the default action specified within the current scope. The current scope is either the current group containing the activity or the orchestration process if the activity is not within a group. • Bypass activity if false — skip the processing of the activity, set the outgoing transition conditions to false without evaluating them, and resume processing. • Throw fault if false — throw a joinFailure fault and switch control to an error-handling routine configured to catch all unhandled exceptions. If the exception is not caught, the fault is propagated to the calling environment.
Join Expression	The expression to evaluate to determine if the activity should be executed. The only data available to this expression is the value of any transitions into the activity. You can use the XPath Formula Builder to create the join expression.

Output

The output for the activity is the following.

Output Item	Datatype	Description
output	varies	The output of this activity is the incoming message from the calling client or service. The structure of the message is determined by the schema of the input message for the selected operation on the Configuration tab.

Output Variable

The output variable tab allows you to select a process variable to contain the output of the activity. The schema of the process variable must match the schema of the output for the activity shown on the Output tab.

Reply With Fault

Activity

The Reply With Fault activity sends a fault message as a reply to a previously received message. This activity is used to respond to the message that an error condition has occurred.

Configuration

The Configuration tab has the following fields.

Field	Description
Name	The name to appear as the label for the activity.
Description	Short description of the activity.
Configuration	Reply messages can be configured in one of the following ways: • Quick configuration — If the reply message is a response to only one message receiver, this option allows you to quickly configure the reply by selecting the activity that received the message. The Reply Event field is used by this option to specify the activity that sent the message to which you wish to reply. • Full configuration — If the reply message can apply to more than one message receiver, this option allows you to specify the Input Partner, Operation, and Message Exchange fields to determine where the reply should be sent. This option allows you to send a reply to a specific receiver, but more than one activity in the process could have received the original message.
Reply Event	This field is available only when Quick configuration is selected in the Configuration field. This field is a drop-down list of the activities in the process that can receive messages (for example, Receive Starter, Receive, On Event, etc.). Select the activity that received the message that you wish to reply to.

Field	Description
Input Partner	This field is available only when Full configuration is selected in the Configuration field. This field specifies the input partner that describes the received message. Together with the Operation and Message Exchange fields, these fields determine where to send the reply message.
Operation	This field is available only when Full configuration is selected in the Configuration field. This field specifies the operation that was invoked by the sender of the message. Together with the Input Partner and Message Exchange field, these fields determine where to send the reply message.
Message Exchange	This field is available only when Full configuration is selected in the Configuration field. This field specifies a string that matches the string specified in the Message Exchange field of the activity that received the message. Together with the Input Partner and Operation fields, these fields determine where to send the reply message.
Fault Name	The fault you wish to send in the reply message. The drop-down list of faults will include all of the fault messages that any activity in the orchestration process can throw.
Correlations	Correlations for the reply message. See Correlations on page 62 for more information about correlations.

Join Condition

The Join Condition tab allows you to define an expression based on the values of the incoming transitions to the activity. If the expression evaluates to true, the activity is executed, if the expression evaluates to false, either the activity is bypassed or a fault is thrown. See Transitions and Join Conditions on page 60 for more information about join conditions.

The Join Condition tab has the following fields.

Field	Description
Join Action	Specifies what action to perform if the join expression evaluates to false. The following options are available: • Inherit setting — use the default action specified within the current scope. The current scope is either the current group containing the activity or the orchestration process if the activity is not within a group. • Bypass activity if false — skip the processing of the activity, set the outgoing transition conditions to false without evaluating them, and resume processing. • Throw fault if false — throw a joinFailure fault and switch control to an error-handling routine configured to catch all unhandled exceptions. If the exception is not caught, the fault is propagated to the calling environment.
Join Expression	The expression to evaluate to determine if the activity should be executed. The only data available to this expression is the value of any transitions into the activity. You can use the XPath Formula Builder to create the join expression.

Input Variable

The Input Variable tab allows you to specify whether you will use the contents of a process variable as input to this activity or you will use the Input tab to specific input bindings as XPath expressions. If you choose to use a process variable, the schema of the variable must match the input schema of the activity exactly.

Field	Description
Input Style	This field allows you to select one of the following input types: • Use Input Variable — specifies that you will select an input variable to use as input for the activity. The schema of the variable must match the input schema of the activity or an error is thrown when this activity is executed. • Use Input Binding Tab — specifies that you will use the Input tab to create mappings and XPath expressions to provide input for this activity.
Input Variable	When Use Input Variable is selected in the Input Style field, this field allows you to select from a list of process variables to use as the input for the activity.

Input

The Input tab is only enabled if Use Input Binding Tab is selected in the Input Style field on the Input Variable tab. See *TIBCO ActiveMatrix BusinessWorks Process Design Guide* for more information about mapping and transforming input data.

The input for the activity is the following.

Input Item	Datatype	Description
input	varies	The reply message to send. The schema depends upon the message defined for the Fault Name selected on the Configuration tab.

Reply

Activity

The Reply activity sends a reply message to a previously received message. This activity is used to send the one of the output messages declared on the input partners of the orchestration process.

Configuration

The Configuration tab has the following fields.

Field	Description
Name	The name to appear as the label for the activity.
Description	Short description of the activity.
Configuration	Reply messages can be configured in one of the following ways: • Quick configuration — If the reply message is a response to only one message receiver, this option allows you to quickly configure the reply by selecting the activity that received the message. The Reply Event field is used by this option to specify the activity that sent the message to which you wish to reply. • Full configuration — If the reply message can apply to more than one message receiver, this option allows you to specify the Input Partner, Operation, and Message Exchange fields to determine where the reply should be sent. This option allows you to send a reply to a specific receiver, but more than one activity in the process could have received the original message.
Reply Event	This field is available only when Quick configuration is selected in the Configuration field. This field is a drop-down list of the activities in the process that can receive messages (for example, Receive Starter, Receive, On Event, etc.). Select the activity that received the message that you wish to reply to.

Field	Description
Input Partner	This field is available only when Full configuration is selected in the Configuration field. This field specifies the input partner that describes the received message. Together with the Operation and Message Exchange fields, these fields determine where to send the reply message.
Operation	This field is available only when Full configuration is selected in the Configuration field. This field specifies the operation that was invoked by the sender of the message. Together with the Input Partner and Message Exchange field, these fields determine where to send the reply message.
Message Exchange	This field is available only when Full configuration is selected in the Configuration field. This field specifies a string that matches the string specified in the Message Exchange field of the activity that received the message. Together with the Input Partner and Operation fields, these fields determine where to send the reply message.
Correlations	Correlations for the reply message. See Correlations on page 62 for more information about correlations.

Join Condition

The Join Condition tab allows you to define an expression based on the values of the incoming transitions to the activity. If the expression evaluates to true, the activity is executed, if the expression evaluates to false, either the activity is bypassed or a fault is thrown. See Transitions and Join Conditions on page 60 for more information about join conditions.

The Join Condition tab has the following fields.

Field	Description
Join Action	Specifies what action to perform if the join expression evaluates to false. The following options are available: • Inherit setting — use the default action specified within the current scope. The current scope is either the current group containing the activity or the orchestration process if the activity is not within a group. • Bypass activity if false — skip the processing of the activity, set the outgoing transition conditions to false without evaluating them, and resume processing. • Throw fault if false — throw a joinFailure fault and switch control to an error-handling routine configured to catch all unhandled exceptions. If the exception is not caught, the fault is propagated to the calling environment.
Join Expression	The expression to evaluate to determine if the activity should be executed. The only data available to this expression is the value of any transitions into the activity. You can use the XPath Formula Builder to create the join expression.

Input Variable

The Input Variable tab allows you to specify whether you will use the contents of a process variable as input to this activity or you will use the Input tab to specific input bindings as XPath expressions. If you choose to use a process variable, the schema of the variable must match the input schema of the activity exactly.

Field	Description
Input Style	This field allows you to select one of the following input types: • Use Input Variable — specifies that you will select an input variable to use as input for the activity. The schema of the variable must match the input schema of the activity or an error is thrown when this activity is executed. • Use Input Binding Tab — specifies that you will use the Input tab to create mappings and XPath expressions to provide input for this activity.
Input Variable	When Use Input Variable is selected in the Input Style field, this field allows you to select from a list of process variables to use as the input for the activity.

Input

The Input tab is only enabled if Use Input Binding Tab is selected in the Input Style field on the Input Variable tab. See *TIBCO ActiveMatrix BusinessWorks Process Design Guide* for more information about mapping and transforming input data.

The input for the activity is the following.

Input Item	Datatype	Description
input	varies	The reply message to send. The schema depends upon the output message of the operation for this reply.

Rethrow

Activity

The Rethrow activity throws the exception caught by the Catch activity again. Use this activity when you wish to propagate the exception to the next level. See Chapter 6, Exception Handling, on page 85 for more information about using the Rethrow activity.

Configuration

The Configuration tab has the following fields.

Field	Description
Name	The name to appear as the label for the activity.
Description	Short description of the activity.

Join Condition

The Join Condition tab allows you to define an expression based on the values of the incoming transitions to the activity. If the expression evaluates to true, the activity is executed, if the expression evaluates to false, either the activity is bypassed or a fault is thrown. See Transitions and Join Conditions on page 60 for more information about join conditions.

The Join Condition tab has the following fields.

Field	Description
Join Action	Specifies what action to perform if the join expression evaluates to false. The following options are available: • Inherit setting — use the default action specified within the current scope. The current scope is either the current group containing the activity or the orchestration process if the activity is not within a group. • Bypass activity if false — skip the processing of the activity, set the outgoing transition conditions to false without evaluating them, and resume processing. • Throw fault if false — throw a joinFailure fault and switch control to an error-handling routine configured to catch all unhandled exceptions. If the exception is not caught, the fault is propagated to the calling environment.
Join Expression	The expression to evaluate to determine if the activity should be executed. The only data available to this expression is the value of any transitions into the activity. You can use the XPath Formula Builder to create the join expression.

Service

Resource

A Service describes the following:

- one or more interfaces and their associated operations.
- endpoint bindings that expose the service to potential clients or partner services.
- partner bindings that associate partner portTypes with concrete bindings so that partner services can be exchanged as needed.

The Service resource has a slightly different set of configuration fields depending on whether you are configuring a TIBCO ActiveMatrix BusinessWorks service or a ActiveMatrix BusinessWorks BPEL Extension service. The Implementation Type field is used to specify which type of service you are configuring. This section describes the configuration of a ActiveMatrix BusinessWorks BPEL Extension service. See *TIBCO ActiveMatrix BusinessWorks Palette Reference* for more information on configuring TIBCO ActiveMatrix BusinessWorks services.

Before creating a service, you will need to create the following:

- WSDL resources for the interfaces the service publishes
- WSDL resources for any partner services that operations can invoke
- Orchestration processes that provide the implementation of the operations
- HTTP Connection or JMS Connection shared configuration resources for the transport used by the endpoints
- Partner Link Configuration resources for any service partners that your orchestration processes invoke.

During deployment, Service resources must be added to a Process Archive. Although it is possible to add a Service resource to a Shared Archive, the resource will not properly function when located in a Shared Archive. Add your Service resources only to Process Archives.

Also, when creating your Enterprise Archive file, the menu item Tools > Create Project EAR does not add Service resources to the Process Archive. You must add any Service resources to the Process Archive by selecting the Processes tab of the Process Archive and clicking the Add a Process Starter to This Archive button.

See Chapter 7, Services and Partners, on page 91 for more information about defining services.

Configuration

The Configuration tab has the following fields.

Field	Description
Name	The name to appear as the label for the resource.
Description	Short description of the resource.
Implementation Type	This field specifies which type of Service this resource uses. The configuration of TIBCO ActiveMatrix BusinessWorks services is slightly different from ActiveMatrix BusinessWorks BPEL Extension services. If you are configuring a ActiveMatrix BusinessWorks BPEL Extension service, select ActiveMatrix BusinessWorks BPEL Extension in this field. If you are configuring a TIBCO ActiveMatrix BusinessWorks service, select ActiveMatrix BusinessWorks Process Definition in this field. See *TIBCO ActiveMatrix BusinessWorks Palette Reference* for more information on configuring TIBCO ActiveMatrix BusinessWorks services.
Implementation	This field contains a set of tabs, one for each interface of the service. Use the + or X button to add or remove interfaces from this field. Adding an interface brings up the Select a Resource dialog where you can select the WSDL resource that describes the correct interface. Use the Rename Current Interface button to rename the currently selected interface.
Interface Tabs Each interface you add is displayed on a tab in the Implementation field. Select the tab of the interface you wish to configure.	
Interface Namespace	Namespace for the WSDL file of the interface. You can click the Go To Refinanced Resource button to view the WSDL file.

Field	Description
Operations Sub-Tab This sub-tab contains a table of all operations defined in the interface. This table allows you to specify the orchestration process that implements each operation. Use the Adjust for WSDL Updates button when the WSDL resource changes.	
Operation Name	Name of the operation. This column is a read-only list of operations contained in the interface.
Timeout (seconds)	Timeout (in seconds) for incoming messages that are not consumed. For example, an incoming message may be for an existing orchestration process, but the orchestration process has not yet reached the point where it is ready to receive the message. Also, you may have incoming unsolicited messages that do not match any existing orchestration process. You can handle unsolicited messages by creating a TIBCO ActiveMatrix BusinessWorks automated process definition to handle them.
Operation Implementation	Double click on this field and use the Browse button to locate the orchestration process that implements each operation. Only orchestration processes whose Input Partners match the operation can be selected.
Endpoint Bindings Sub-Tab This sub-tab allows you to specify one or more endpoints that expose the service as well as the transport bindings for each endpoint. Use the + button to add endpoints, use the X button to delete endpoints, and use the arrow buttons to move the selected endpoint in the list.	
Endpoint Name	Double click this field to specify a name for this endpoint.

Field	Description
Endpoint Type	Specify the type of endpoint by selecting from the list of supported types: • Local — A local endpoint is available to consumers within the same process engine. This type of endpoint provides a highly efficient way for local services to invoke each other without incurring the overhead of using a network transport. • SOAP — A SOAP endpoint is used to expose the service by way of the SOAP protocol to other partner services or web clients.
Transport Sub-Tab	
Transport	Use the Browse button to select a transport. HTTP Connection or JMS Connection resources can be used as transports.
Use Basic Authentication (HTTP Transport)	This field is available when an HTTP transport is selected. When checked, this field specifies that incoming SOAP requests must supply a valid username and password. The user name and password specified in the incoming request must exist in the domain (users are created and managed in the domain using TIBCO Administrator).
Endpoint URI (HTTP Transport)	This field is available when an HTTP transport is selected. This field specifies the Endpoint URI that clients can use to access the service.

Field	Description
JMS Destination (JMS Transport)	This field is available when a JMS transport is selected. This field specifies the name of the destination for incoming JMS messages for this service. The syntax of the destination name is specific to the JMS provider you are using. See your JMS provider documentation for more information about destination names. **Note:** If you are using TIBCO Enterprise Message Service as your JMS provider, you can use the Browse button next to this field after specifying a valid connection in the JMS Connection field. The Browse button displays a list of configured destinations in the JMS server that are appropriate for this activity.
JMS Destination Type (JMS Transport)	This field is available when a JMS transport is selected. This field specifies whether the JMS Destination is a Topic or a Queue.
JMS Message Type (JMS Transport)	This field is available when a JMS transport is selected. This field specifies the type of incoming messages on the specified destination. Can be either Text Message or Bytes message.

Field	Description
Acknowledge Mode (JMS Transport)	This field is available when a JMS transport is selected. This field specifies the acknowledgement mode to use for incoming messages. Can be one of the following: • Auto — the message is automatically acknowledged when it is received. • Client — the message will be acknowledged when the process implementing the operation ends successfully. • Dups OK — the message is acknowledged automatically when it is received. JMS provides this mode for lazy acknowledgement, but TIBCO ActiveMatrix BusinessWorks acknowledges messages upon receipt. • TIBCO EMS Explicit Client Acknowledge — (only available for TIBCO EMS) the message will be acknowledged at a later point by using the Confirm activity. The session is not blocked and one session handles all incoming messages for each process instance. If a message is not confirmed before the process instance ends, all messages received in the same session are redelivered.

Field	Description
Max Sessions (JMS Transport)	This field is available when a JMS transport is selected and Client is selected as the Acknowledge Mode. If the JMS Destination Type is Topic, this field is read-only. If the JMS Destination Type is Queue, the value of this field can be altered. This field specifies the maximum number of JMS sessions to create for incoming queue messages. When a JMS queue message is received, the session is blocked until the message is acknowledged. Because the acknowledgement can comes at a later time when the process ends, this field allows you to specify a maximum number of new sessions to create to handle incoming messages. Once the maximum number of sessions is reached, no new incoming messages can be processed. Once an incoming message is confirmed, the total number of active sessions is decreased and another incoming message can be processed.
Durable Subscription (JMS Transport)	This field is available when a JMS transport is selected and Topic is selected as the JMS destination type. This field allows you to specify whether each operation requires a durable topic subscription.
SOAP Details Sub-Tab	
Default Style	The SOAP binding style for operations that do not explicitly set their binding style. You can specify either Document or RPC style.
SOAP Version	Specify either SOAP 1.1 or SOAP 1.2 for the version of the SOAP specification to which incoming messages should comply.

Field	Description
Operations	Select each operation from the drop down list to specify the SOAP Action, Style, or Encoding for the operation. The Advanced button allows advanced configuration of SOAP headers on BPEL endpoints for inbound service request. These headers can be mapped into a context object which is made available as service context in the orchestration through the Get Context activity. See, Advanced SOAP Settings, on page 177.
Soap Action	The soapAction that is expected from incoming SOAP requests. This field is required, and by default, the operation name is used as the soapAction. Each operation in an interface must have a unique value for SOAP Action. See the SOAP specification for more information about soapAction.
Style	The SOAP binding style for the selected operation. You can specify either Document or RPC style, or you can specify that the value in the Default Style field should be used.
Encoding	The encoding type for the body of the SOAP input and output messages. This can be either literal or encoded. Encoded messages support more complex datatypes such as SOAP arrays. When encoded is specified, you can optionally specify the namespace for input, output, and fault messages.
Input Message Namespace	The namespace for input messages.
Output Message Namesapce	The namespace for output messages.
Fault Message Namespace	The namespace for fault messages.

Advanced SOAP Settings

For each operation, you can specify advanced configuration options. To access the advanced configuration options, perform the following:

1. Click the Endpoint Bindings Tab.
2. Select the required endpoint and click the SOAP Details sub-tab.
3. Select the operation in the Operations field.
4. Click the **Advanced** button next to the operation name.

The Advanced SOAP Settings dialog appears. The tabs on the Advanced SOAP Settings dialog are the same as the dialog in TIBCO ActiveMatrix BusinessWorks. Refer to Chapter 16, "Service Palette" of the *TIBCO ActiveMatrix BusinessWorks Palette Reference* for more information on the tabs of this dialog.

WSDL Source

The WSDL Source tab displays the concrete WSDL interface file that is generated based on the information specified on the Configuration tab of this resource. This tab has the following fields.

Field	Global Var?	Description
Service URI	No	The Service URI portion of the URL that can be used to retrieve the WSDL file. For consumers to retrieve WSDL files from TIBCO ActiveMatrix BusinessWorks, you must define a process definition that accepts HTTP requests and uses the Retrieve Resources activity to generate the WSDL file. See *TIBCO ActiveMatrix BusinessWorks Palette Reference* for more information. The URL to retrieve WSDL files is the following: `http://<host>:<port>/<serviceURI>?wsdl` where <*serviceURI*> is the value in this field.
Target Namespace	No	Target namespace of the Service.

Field	Global Var?	Description
Embed In WSDL	No	Check one or more of the following to embed the selected item(s) in the concrete WSDL file instead of using import statements: • Interface — When checked, specifies that all referenced WSDL files should be included inline in the concrete WSDL file. Otherwise, all referenced WSDL files are imported. Do not use this option if you plan on distributing the concrete WSDL to other services. • Types — When checked, specifies that all referenced XSD files should be included inline in the concrete WSDL file. Otherwise, all referenced XSD files are imported. • JNDI Properties — When checked, specifies that JDNI properties should be included inline in the service specification in the WSDL Source tab.
WSDL Code	No	This field displays the concrete WSDL file that describes this service. Click the Save WSDL File button to save the concrete WSDL file to disk, if desired.

Overview

The Overview tab provides a tree view of the service that you can expand and collapse to view the exposed endpoints and operations for the service.

Sleep

Activity

The Sleep activity suspends the process on the current transition for the given amount of time. If you have multiple control flows in your process, only the current execution branch of the process is suspended.

One use of the Sleep activity is to set a timeout for a Pick First group. See Pick First Groups on page 77 for an example of using Sleep within a Pick First group.

Configuration

The Configuration tab has the following fields.

Field	Description
Name	The name to appear as the label for the activity.
Description	Short description of the activity.
Style	Specifies the style of determining when to end the sleep period. Can be one of the following: • For — specifies an amount of time in years, days, months, hours, minutes, seconds, and milliseconds to sleep. • Until — specifies a date until which to sleep.
For Config Option	You can specify one of the following: • Literal — this allows you to provide literal values, such as 1 day or 3 months. • Expression — this allows you to provide an XPath expression that evaluates to an amount of time.
Until Config Option	You can specify one of the following: • Literal — this allows you to choose a specific date and time from a calendar and clock. • Expression — this allows you to provide an XPath expression that evaluates to a specific date and time.

Field	Description
For	Specify the amount of time in the provided fields: Years, Months, Days, Hours, Minutes, Seconds, Milliseconds.
For Expression	Specify an XPath expression that evaluates to an amount of time. Use the XPath Formula Builder button to bring up the formula editor, if desired.
Until	Specify a specific date using the calendar and clock provided.
Until Expression	Specify an XPath expression that evaluates to a specific date and time. Use the XPath Formula Builder to bring up the formula editor, if desired.

Join Condition

The Join Condition tab allows you to define an expression based on the values of the incoming transitions to the activity. If the expression evaluates to true, the activity is executed, if the expression evaluates to false, either the activity is bypassed or a fault is thrown. See Transitions and Join Conditions on page 60 for more information about join conditions.

The Join Condition tab has the following fields.

Field	Description
Join Action	Specifies what action to perform if the join expression evaluates to false. The following options are available: • Inherit setting — use the default action specified within the current scope. The current scope is either the current group containing the activity or the orchestration process if the activity is not within a group. • Bypass activity if false — skip the processing of the activity, set the outgoing transition conditions to false without evaluating them, and resume processing. • Throw fault if false — throw a joinFailure fault and switch control to an error-handling routine configured to catch all unhandled exceptions. If the exception is not caught, the fault is propagated to the calling environment.
Join Expression	The expression to evaluate to determine if the activity should be executed. The only data available to this expression is the value of any transitions into the activity. You can use the XPath Formula Builder to create the join expression.

Throw

Activity

The Throw activity throws the specified fault and control is passed to any error-handling routine defined to handle the error. If no error-handling routine is defined in the orchestration process, the error is propagated to the calling environment (for example, the client or service that sent the request). See Chapter 6, Exception Handling, on page 85 for more information about error handling.

Configuration

The Configuration tab has the following fields.

Field	Description
Name	The name to appear as the label for the activity.
Description	Short description of the activity.
Fault Namespace	Namespace to use for the fault.
Fault Local Name	Name of the fault. You can define any fault to throw. The schema of the fault is defined on the Fault Data tab.

Fault Data

The Fault Data tab defines the schema for the fault you wish to throw. If you choose to throw a pre-defined fault message for one of the incoming messages in the process, you should select the correct fault message from the WSDL resource. You can also choose to throw a new fault and define your own fault schema.

See *TIBCO ActiveMatrix BusinessWorks Process Design Guide* for more information about defining schemas.

Join Condition

The Join Condition tab allows you to define an expression based on the values of the incoming transitions to the activity. If the expression evaluates to true, the activity is executed, if the expression evaluates to false, either the activity is bypassed or a fault is thrown. See Transitions and Join Conditions on page 60 for more information about join conditions.

The Join Condition tab has the following fields.

Field	Description
Join Action	Specifies what action to perform if the join expression evaluates to false. The following options are available: • Inherit setting — use the default action specified within the current scope. The current scope is either the current group containing the activity or the orchestration process if the activity is not within a group. • Bypass activity if false — skip the processing of the activity, set the outgoing transition conditions to false without evaluating them, and resume processing. • Throw fault if false — throw a joinFailure fault and switch control to an error-handling routine configured to catch all unhandled exceptions. If the exception is not caught, the fault is propagated to the calling environment.
Join Expression	The expression to evaluate to determine if the activity should be executed. The only data available to this expression is the value of any transitions into the activity. You can use the XPath Formula Builder to create the join expression.

Input Variable

The Input Variable tab allows you to specify whether you will use the contents of a process variable as input to this activity or you will use the Input tab to specific input bindings as XPath expressions. If you choose to use a process variable, the schema of the variable must match the input schema of the activity exactly.

Field	Description
Input Style	This field allows you to select one of the following input types: • Use Input Variable — specifies that you will select an input variable to use as input for the activity. The schema of the variable must match the input schema of the activity or an error is thrown when this activity is executed. • Use Input Binding Tab — specifies that you will use the Input tab to create mappings and XPath expressions to provide input for this activity.
Input Variable	When Use Input Variable is selected in the Input Style field, this field allows you to select from a list of process variables to use as the input for the activity.

Input

The Input tab is only enabled if Use Input Binding Tab is selected in the Input Style field on the Input Variable tab. See *TIBCO ActiveMatrix BusinessWorks Process Design Guide* for more information about mapping and transforming input data.

The input for the activity is the following.

Input Item	Datatype	Description
input	varies	The schema for the fault message to throw. The schema depends upon the fault message.

Write To Log

Activity

This activity writes a message to the log.

Depending upon whether you are using a process engine in testing mode or a deployed process engine, the logs are stored in different locations. During test mode, the logs are stored in the TIBCO ActiveMatrix BusinessWorks working directory under the `logs` subdirectory. See *TIBCO Designer User's Guide* for more information about specifying the working directory. For deployed process engines, the log location is specified by custom engine properties. See *TIBCO ActiveMatrix BusinessWorks Administration* for more information about custom engine properties.

Configuration

The Configuration tab has the following fields.

Field	Description
Name	The name to appear as the label for the activity.
Description	Short description of the activity.
Role	The user-defined role name for this log entry. Roles can be used to filter the log entries when displaying them. For example, if you set the role as Debug, you can choose to view or not view all log entries that have the role set to Debug. You can enable or disable system or user-defined roles using custom engine properties at deployment time. See *TIBCO ActiveMatrix BusinessWorks Administration* for more information.
Suppress Job Info	If checked, no additional information is added to the log entry. If unchecked, each log message has the Job number, process definition name, and activity name prepended to the message text.

Join Condition

The Join Condition tab allows you to define an expression based on the values of the incoming transitions to the activity. If the expression evaluates to true, the activity is executed, if the expression evaluates to false, either the activity is bypassed or a fault is thrown. See Transitions and Join Conditions on page 60 for more information about join conditions.

The Join Condition tab has the following fields.

Field	Description
Join Action	Specifies what action to perform if the join expression evaluates to false. The following options are available: • Inherit setting — use the default action specified within the current scope. The current scope is either the current group containing the activity or the orchestration process if the activity is not within a group. • Bypass activity if false — skip the processing of the activity, set the outgoing transition conditions to false without evaluating them, and resume processing. • Throw fault if false — throw a joinFailure fault and switch control to an error-handling routine configured to catch all unhandled exceptions. If the exception is not caught, the fault is propagated to the calling environment.
Join Expression	The expression to evaluate to determine if the activity should be executed. The only data available to this expression is the value of any transitions into the activity. You can use the XPath Formula Builder to create the join expression.

Input

See *TIBCO ActiveMatrix BusinessWorks Process Design Guide* for more information about mapping and transforming input data.

The input for the activity is the following.

Input Item	Datatype	Description
message	string	The message that you would like to appear in the log. Logs are stored in the TIBCO ActiveMatrix BusinessWorks installation directory under the `logs` subdirectory. **Note:** When the message contains non-ASCII data, the default encoding of the Java Virtual Machine used by the TIBCO ActiveMatrix BusinessWorks process engine is used to encode the text when writing to the log file.
msgCode	string	The error code of the message. This is used as an identifier for the message so that applications can scan the log for the message.

Appendix A Error Codes

This appendix lists the error codes that can be encountered when using TIBCO ActiveMatrix BusinessWorks BPEL Extension.

Topics

TIBCO ActiveMatrix BusinessWorks BPEL Extension Error Codes

SC-BWBPEL-100001: [Service = {0}; Implementation = {1}; Operation = {2}]. The implementation references an invalid Input Partner [{3}].

Role: errorRole

Category: SC-BW-BPEL

Description: [Service = {0}; Implementation = {1}; Operation = {2}]. The implementation references an invalid Input Partner [{3}].

Resolution: Make sure the input partner in the orchestration process is configured correctly.

SC-BWBPEL-100002: [Service = {0}]: The service references an invalid partner [{1}]. Unable to resolve partner.

Role: errorRole

Category: SC-BW-BPEL

Description: [Service = {0}]: The service references an invalid partner [{1}]. Unable to resolve partner.

Resolution: Make sure the partner in the orchestration process is configured correctly.

SC-BWBPEL-100003: [Partner Link Configuration = {0}; Partner Link = {1}]. Partner Link has an invalid timeout [{2}].

Role: errorRole

Category: SC-BW-BPEL

Description: [Partner Link Configuration = {0}; Partner Link = {1}]. Partner Link has an invalid timeout [{2}].

Resolution: Make sure timeout is numeric.

SC-BWBPEL-100004: [Partner Link Configuration = {0}; Partner Link = {1}]. Partner Link has invalid configuration.

Role: errorRole

Category: SC-BW-BPEL

Description: [Partner Link Configuration = {0}; Partner Link = {1}]. Partner Link has invalid configuration.

Resolution: Depends on content of exception

SC-BWBPEL-100005: [Service = {0}; Implementation = {1}; Operation = {2}]. Timeout value is invalid [{3}].

Role: errorRole

Category: SC-BW-BPEL

Description: [Service = {0}; Implementation = {1}; Operation = {2}]. Timeout value is invalid [{3}].

Resolution: Make sure timeout is numeric.

SC-BWBPEL-100006: [Partner Link Configuration = {0}]. Partner Link name is not specified.

Role: errorRole

Category: SC-BW-BPEL

Description: [Partner Link Configuration = {0}]. Partner Link name is not specified.

Resolution: Specify a partner link name.

SC-BWBPEL-100007: [Partner Link Configuration = {0}; Partner Link = {1}]. Endpoint is not found.

Role: errorRole

Category: SC-BW-BPEL

Description: [Partner Link Configuration = {0}; Partner Link = {1}]. Endpoint is not found.

Resolution: Specify a valid endpoint.

SC-BWBPEL-100008: [Service = {0}]. Partner Link is not found.

Role: errorRole

Category: SC-BW-BPEL

Description: [Service = {0}]. Partner Link is not found.

Resolution: Make sure partner link exists.

SC-BWBPEL-100009: [Service = {0}]. Partner Link is invalid.

Role: errorRole

Category: SC-BW-BPEL

Description: [Service = {0}]. Partner Link is invalid.

Resolution: Make sure partner link exists and configured correctly.

SC-BWBPEL-100010: Unable to locate reference [{0}].

Role: errorRole

Category: SC-BW-BPEL

Description: Unable to locate reference [{0}].

Resolution: Correct use of reference

SC-BWBPEL-100011: [Service = {0}; Partner Link Configuration = {1}; Partner Link = {2}]. A partner binding cannot refer to the service that contains it.

Role: errorRole

Category: SC-BW-BPEL

Description: [Service = {0}; Partner Link Configuration = {1}; Partner Link = {2}]. A partner binding cannot refer to the service that contains it.

Resolution: Do not use the current service in its own partner bindings. Correct the partner binding to point to a different service.

BWBPEL-100001: Support exception

Role: errorRole

Category: BW-BPEL

Description: Support generate exception

Resolution: Depends on content of exception

BWBPEL-100002: Input variable not specified

Role: errorRole

Category: BW-BPEL

Description: Input variable name not specified

Resolution: Specify input variable

BWBPEL-100003: Cannot reply, missing request [%1]

Role: errorRole

Category: BW-BPEL

Description: Can not find event context matching this key

Resolution: Match reply with receive

BWBPEL-100004: Send reply error

Role: errorRole

Category: BW-BPEL

Description: Send reply failed

Resolution: Depends on content of exception

BWBPEL-100005: Conflicting receive starter activities

Role: errorRole

Category: BW-BPEL

Description: Multiple receive starter activities are not allowed for the same message

Resolution:

BWBPEL-100006: Conflicting outstanding request [%1]

Role: errorRole

Category: BW-BPEL

Description: Conflicting outstanding reply pending for same partner, operation and message exchange

Resolution:

BWBPEL-100007: Missing reply [%1]

Role: errorRole

Category: BW-BPEL

Description: Process has ended without a reply to an outstanding request

Resolution:

BWBPEL-100008: Conflicting concurrent receive activities

Role: errorRole

Category: BW-BPEL

Description: Multiple concurrent receive activities for the same partner, opertation and correlation set

Resolution: Resolve conflict

BWBPEL-100009: Correlation previously set [%1]

Role: errorRole

Category: BW-BPEL

Description: Yes correlation should not have been previously set

Resolution: Correct use of correlations

BWBPEL-100010: Correlation not previously set [%1]

Role: errorRole

Category: BW-BPEL

Description: No correlation should have been previously set

Resolution: Correct use of correlations

BWBPEL-100011: Correlation does not match [%1]

Role: errorRole

Category: BW-BPEL

Description: Correlation should have matched previously set value

Resolution: Correct use of correlations

BWBPEL-100012: Correlation name not unique [%1]

Role: errorRole

Category: BW-BPEL

Description: Correlation name not unique

Resolution: Make correlation names unique

BWBPEL-100013: Fault name not specified

Role: errorRole

Category: BW-BPEL

Description: Fault name to Throw not specified

Resolution: Specify Fault local name to Throw

BWBPEL-100014: Fault data specification error [%1]

Role: errorRole

Category: BW-BPEL

Description: Error with Fault Data specification

Resolution: Correct Fault Data specification

BWBPEL-100015: Missing Expression

Role: errorRole

Category: BW-BPEL

Description: Missing Expression

Resolution: Specify missing expression

BWBPEL-100016: Assignment error for from [%1] to [%2]

Role: errorRole

Category: BW-BPEL

Description: Assignment error encountered

Resolution: Correct Assign expressions

BWBPEL-100017: Assignment error creating new Variable: for from [%1] to [%2]

Role: errorRole

Category: BW-BPEL

Description: Assignment error encountered creating new variable

Resolution: Correct Assign expression

BWBPEL-100018: Invalid dateTime value [%1]

Role: errorRole

Category: BW-BPEL

Description: Incorrect value for dateTime format

Resolution: Use valid value

BWBPEL-100019: Invalid duration value [%1]

Role: errorRole

Category: BW-BPEL

Description: Incorrect value for duration

Resolution: Use valid value

BWBPEL-100020: Duration value not positive [%1]

Role: errorRole

Category: BW-BPEL

Description: Incorrect value for duration

Resolution: Use valid value

BWBPEL-100021: Fault occurred

Role: errorRole

Category: BW-BPEL

Description: Fault occurred

Resolution: Take action based on the fault

BWBPEL-100022: Assign From expression [%1] error

Role: errorRole

Category: BW-BPEL

Description: Assign From XPath expression has an error

Resolution: Correct XPath expression

BWBPEL-100023: Assign To expression [%1] error

Role: errorRole

Category: BW-BPEL

Description: Assign To XPath expression has an error

Resolution: Correct XPath expression

BWBPEL-100024: Invoke Message Correlation error

Role: errorRole

Category: BW-BPEL

Description: Invoke Message Correlation error, see message for details

Resolution: Correct Correlation

BWBPEL-100025: Reply Message Correlation error

Role: errorRole

Category: BW-BPEL

Description: Reply Message Correlation error, see message for details

Resolution: Correct Correlation

BWBPEL-100026: Invoke Message Correlation expression [%1] error

Role: errorRole

Category: BW-BPEL

Description: Correlation XPath expression has an error for Invoke Message

Resolution: Correct XPath expression

BWBPEL-100027: Reply Message Correlation expression [%1] error

Role: errorRole

Category: BW-BPEL

Description: Correlation XPath expression has an error for Reply Message

Resolution: Correct XPath expression

BWBPEL-100028: Correlation error

Role: errorRole

Category: BW-BPEL

Description: Correlation error, see message for details

Resolution: Correct Correlation

BWBPEL-100029: Correlation expression [%1] error

Role: errorRole

Category: BW-BPEL

Description: Correlation XPath expression has an error

Resolution: Correct XPath expression

BWBPEL-100030: Correlation expression not specified for [%1]

Role: errorRole

Category: BW-BPEL

Description: Correlation expression not specified this correlation

Resolution: Specify correlation expression

BWBPEL-200001: Assign From expression [%1] warning

Role: warnRole

Category: BW-BPEL

Description: Assign From XPath expression has a warning

Resolution: Verify XPath expression

BWBPEL-200002: Assign To expression [%1] warning

Role: warnRole

Category: BW-BPEL

Description: Assign To XPath expression has a warning

Resolution: Verify XPath expression

BWBPEL-200003: Invoke Message Correlation expression [%1] warning

Role: warnRole

Category: BW-BPEL

Description: Correlation XPath expression has a warning for Invoke Message

Resolution: Verify XPath expression

BWBPEL-200004: Reply Message Correlation expression [%1] warning

Role: warnRole

Category: BW-BPEL

Description: Correlation XPath expression has a warning for Reply Message

Resolution: Verify XPath expression

BWBPEL-200005: Correlation expression [%1] warning

Role: warnRole

Category: BW-BPEL

Description: Correlation XPath expression has a warning

Resolution: Verify XPath expression

BWBPEL-100101: The target namespace is not specified.

Role: errorRole

Category: BW-BPEL

Description: The target namespace is not specified.

Resolution: Specify target namespace for the port type in the Input Partners tab.

BWBPEL-100102: The {0} name is not specified.

Role: errorRole

Category: BW-BPEL

Description: The {0} name is not specified.

Resolution: Specify a value in the Name column of the table in the Partner Link field.

BWBPEL-100103: The {0} name [{1}] is not valid.

Role: errorRole

Category: BW-BPEL

Description: The {0} name [{1}] is not valid.

Resolution: The value in the Name column in the table in the Partner Link field cannot be empty or only whitespace characters.

BWBPEL-100104: The PortType name is not specified in {0} [{1}].

Role: errorRole

Category: BW-BPEL

Description: The PortType name is not specified in {0} [{1}].

Resolution: Specify a port type.

BWBPEL-100105: The PortType Name [{0}] specified in {1} [{2}] is not valid.

Role: errorRole

Category: BW-BPEL

Description: The PortType Name [{0}] specified in {1} [{2}] is not valid.

Resolution: PortType name cannot be empty or only whitespace characters.

BWBPEL-100106: The PortType namespace is not specified in {0} [{1}].

Role: errorRole

Category: BW-BPEL

Description: The PortType namespace is not specified in {0} [{1}].

Resolution: Specify a namespace for the PortType.

BWBPEL-100107: The PortType namespace [{0}] specified in {1} [{2}] is not valid.

Role: errorRole

Category: BW-BPEL

Description: The PortType namespace [{0}] specified in {1} [{2}] is not valid.

Resolution: The PortType Namespace cannot be empty or only whitespace characters.

BWBPEL-100115: Input partners are not specified. An orchestration process must define at least one input partner.

Role: errorRole

Category: BW-BPEL

Description: Input partners are not specified. An orchestration process must define at least one input partner.

Resolution: Specify an input partner.

Index

A

B

C

D

E

G

I

J

L

M

N

O

P

R

S

T

U

V

W