

TIBCO ActiveMatrix BusinessWorks™

Process Design

*Software Release 5.14
August 2018*

Important Information

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE "LICENSE" FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

ANY SOFTWARE ITEM IDENTIFIED AS THIRD PARTY LIBRARY IS AVAILABLE UNDER SEPARATE SOFTWARE LICENSE TERMS AND IS NOT PART OF A TIBCO PRODUCT. AS SUCH, THESE SOFTWARE ITEMS ARE NOT COVERED BY THE TERMS OF YOUR AGREEMENT WITH TIBCO, INCLUDING ANY TERMS CONCERNING SUPPORT, MAINTENANCE, WARRANTIES, AND INDEMNITIES. DOWNLOAD AND USE THESE ITEMS IS SOLELY AT YOUR OWN DISCRETION AND SUBJECT TO THE LICENSE TERMS APPLICABLE TO THEM. BY PROCEEDING TO DOWNLOAD, INSTALL OR USE ANY OF THESE ITEMS, YOU ACKNOWLEDGE THE FOREGOING DISTINCTIONS BETWEEN THESE ITEMS AND TIBCO PRODUCTS.

This document contains confidential information that is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIBCO, Two-Second Advantage, TIBCO Hawk, TIBCO Rendezvous, TIBCO Runtime Agent, TIBCO ActiveMatrix BusinessWorks, TIBCO Administrator, TIBCO Designer, TIBCO ActiveMatrix Service Gateway, TIBCO BusinessEvents, TIBCO BusinessConnect, and TIBCO BusinessConnect Trading Community Management are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

Enterprise Java Beans (EJB), Java Platform Enterprise Edition (Java EE), Java 2 Platform Enterprise Edition (J2EE), and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle Corporation in the U.S. and other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

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. SEE THE README FILE FOR THE AVAILABILITY OF THIS SOFTWARE VERSION ON A SPECIFIC OPERATING SYSTEM PLATFORM.

THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

Copyright © 2001-2018 TIBCO Software Inc. ALL RIGHTS RESERVED.

Contents

Figures	xv
Tables	xix
Preface	xxi
Changes from the Previous Release of this Guide	xxii
Related Documentation	xxiii
TIBCO ActiveMatrix BusinessWorks Documentation	xxiii
Other TIBCO Product Documentation	xxiv
Typographical Conventions	xxv
TIBCO Product Documentation and Support Services	xxvii
How to Access TIBCO Documentation	xxvii
How to Contact TIBCO Support	xxvii
How to Join TIBCO Community	xxvii
Chapter 1 Getting Started With TIBCO Designer	1
Welcome to TIBCO Designer	2
Starting TIBCO Designer	3
Startup Options	3
TIBCO Designer Administration	6
TIBCO Designer Interface Overview	8
Main Window	8
Project Panel	9
Palette Panel	11
Design Panel	12
Configuration Panel	12
Customizing the Display	14
Choosing Panel Layout	14
Choosing Palette Mode or Non-palette Mode	15
Displaying Palettes in a Separate Window	17
Working With User Palettes	17
Accessing Documentation	19

Chapter 2 Managing Projects and Resources	21
Overview of Projects	22
Project Structure	22
Using a Version Control System	23
Project Templates	23
Creating Projects	24
Validating Projects	25
Saving Projects	26
Opening and Reopening Projects	28
Adding Resources To Your Project	29
If Adding a Resource Results in an Error	30
Deleting Projects	31
Tips and Tricks for Working With Projects	32
Chapter 3 Processes	33
Business Process Modeling	34
TIBCO ActiveMatrix BusinessWorks Solves Enterprise Integration Problems	36
Overview of Processes	37
Process Definitions	38
Activities	39
Transitions	40
Groups	41
Shared Configuration Resources	42
Subprocesses	43
Developing Process Definitions	45
Chapter 4 Activities	47
Activity Overview	48
Activity Icons	50
Configuration	52
Advanced	53
Event	54
Editor	56
Input	62
Mapping and Transforming Activity Input Data	62
Output	64
Process Starters	65

Misc Tab	66
Start Activity	67
Configuration	67
Output Editor	67
Output	68
End Activity	69
Configuration	69
Input Editor	69
Input	70
Error Schemas	70
Chapter 5 Transitions and Conditions	71
Transitions	72
Creating a Transition	74
Conditions	75
Chapter 6 Grouping Activities	77
Overview of Groups	78
Activity Output and Groups	80
Group Configuration Tab	82
No Action Groups	88
If Groups	89
Critical Section Groups	90
Synchronization Options	90
Usage Guidelines	91
Pick First Groups	92
Overview of Loops	93
Index Variable	93
Accumulate Output	93
Iterate Loop	95
Iteration Element	96
Repeat Until True Loop	98
While True Loop	100
Repeat On Error Until True Loop	101
Suspend If Still Error Option	102
Chapter 7 Working With Variables	105
Overview of Variables	106
Global Variables	107
Global Variables Editor	107

Changing Global Variable Values at Runtime	110
Process Variables	111
Activity Output	111
Predefined Process Variables	111
Error Process Variables	112
User-Defined Process Variables	112
Memory Usage of Process Variables	113
Shared Variables	114
Shared Variable	114
Job Shared Variable	114
Configuring Shared Variables	114
Assigning and Retrieving the Variable's Value	116
Chapter 8 Mapping and Transforming Data	117
Overview of Mapping and Transformation	118
Process Data Panel	118
Activity Input Panel	119
Mapping and Transforming Process Data to Activity Input	119
Statements, Hints, and Errors	120
Buttons, Menus, and Icons	121
Toolbar and Right-Click Menu on the Input Tab	121
Icons for Schema Element Datatypes	124
Qualifier Icons	125
Specifying Constants	126
Date and Datetime Strings in Constants	126
Data Validation	127
Repairing Incorrect Mappings	130
Shortcuts	132
Right-Click Menu	132
Dragging to the Left	132
Cutting and Pasting	133
Automatic Testing	134
Examples of Mappings	137
Using XPath	137
Setting an Element Explicitly to Nil	137
Merging Input from Multiple Sources	139
Converting a List Into a Grouped List	144
Merging Two Corresponding Lists	148
Coercions	151
XSLT Statements	155
Attribute	155
Choose	155

Comment	156
Copy	156
Copy-Contents-Of	156
Copy-Of	156
Element	157
For-Each	157
For-Each-Group	157
Generate Comment	157
Generate PI	158
If	158
Value-Of	158
Variable	158
Chapter 9 XPath	161
XPath Basics	162
Addressing Schema Elements	162
Evaluation Context	163
Namespaces	163
Search Predicates	164
Testing For Nil	164
Comments	164
The XPath Formula Builder	165
String Representations of Datatypes	168
Date and Time Functions	170
BW Process Information Functions	173
Chapter 10 Error Handling	174
Overview of Error Handling	175
The Error Process Variables	176
\$_error Process Variable	176
\$_error_<activityName> Process Variables	177
Error Propagation	179
Group Error Propagation	179
Called Process Error Propagation	180
Process Error Schemas	181
Using the Catch and Rethrow Activities	183
Chapter 11 Transactions	185
Overview of Transactions	186
Types of Transactions	186
JDBC	188

Multiple JDBC Connections In Transaction Groups	188
Configuring JDBC Transactions	188
Java Transaction API (JTA) UserTransaction	189
Configuring JTA UserTransaction Transactions	190
XATransaction	191
Configuring XA Transactions	191
Configuring Transaction Managers	193
Transaction Recovery	196
PreCreating XA Transactions At Event Sources	198
Monitoring XA Transactions	198
JMS Local Transaction	199
Configuring JMS Local Transactions	199
Limitations while working with JMS Local Transactions	200
Nested Transactions	202
Summary of Transactions	204
Chapter 12 Process Instance Execution	207
Detecting Duplicate Process Instances	208
When to Perform Checkpoints	209
Specifying the Duplicate Key	209
Transactions and Duplicate Detection	210
Handling Duplicate Messages	210
Process Engine Properties for Duplicate Detection	211
Sequencing Process Instances	212
Example 1: Processing Orders As They Are Received	212
Example 2: Periodic Processing	213
Example 3: Handling Client Messages	214
Logging for Third-Party Components	215
Chapter 13 Inter-Process Communication	217
Overview of Inter-Process Communication	218
Data for Inter-Process Communication	220
Coordinating Inter-Process Communication	221
Specifying the Key	221
Timeouts for Notify and Wait	221
Database Storage for Wait/Notify/Receive Notification Information	223
Examples of Inter-Process Communication	224
Enforcing Order for Process Execution	224
Multiple Types of Incoming Events Resume a Running Process	225
Scalability With Incoming Events	226
Specific Protocol Requirements	226

Chapter 14 Invoking and Implementing Web Services	229
Overview of Web Services	230
The Service Palette	232
Context Separation	233
Configuring a Service	233
Operation Implementation	234
Running Services	235
Partners and Partner Link Configurations	236
Applying WS-Security Policy for Partner Link Configuration	237
Associating Policies with the Invoke Partner Activity	239
Headers for Declared Faults	239
Configuring Context Resource	242
The SOAP Palette	245
Sending SOAP Requests	246
Receiving SOAP Requests	246
Processing SOAP Messages with Attachments	247
SOAP Undescribed Headers	252
Partner Link Configuration for Undescribed Header Support	256
SOAP Fault Sub-Element Propagation	258
Fault Context for a Service Resource	258
Configuring Fault Context on Partner Link Configuration	258
Web Service Wizards	260
Creating Web Services From Process Definitions	260
Creating Process Definition Stubs From a WSDL File	262
Using Web Services Security Policies	264
Associating Security Policies With Web Services	265
Custom Password Lookup	265
Security Context Propagation from TIBCO ActiveMatrix Policy Manager	267
Using JAAS Login for Authentication	271
WSIL Files and UDDI Registries	274
Connect	274
Browse	276
Publish	278
Chapter 15 Using the Schema Import Tool	281
Overview of the Schema Import Tool	282
The Location Resource	283
Importing Resources	284
Chapter 16 Working With Secure Sockets Layer (SSL)	285
Overview of SSL	286

Identity Resources	287
Username/ Password	287
Certificate/Private Key	287
Identity File	287
Trusted Certificates	289
Adding Certificates to Your Project	290
Storing Trusted Certificates Outside of Your Project	291
SSL Configuration	293
Chapter 17 Creating Custom Activities and Process Starters	301
Overview of Custom Activities	302
Creating Custom Activities	304
Packaging Custom Activities Into a Custom Palette	306
Using Custom Activities	308
Global Variables in Custom Activities	308
Trace Information and Packaged Processes	308
Custom Process Starters	309
Chapter 18 Testing Process Definitions	311
Overview of Testing	312
Breakpoints	313
The Test Panel	315
Process Instances During Testing	316
Loading Processes to Test	316
Creating Process Instances	316
Working With Process Instances	317
Stepping Through a Process	318
Colors in Test Mode	319
Test Mode Buttons and Menus	320
Chapter 19 Sharing Common Resources with Other Projects	325
Overview	326
AliasLibrary Overview	326
LibraryBuilder Overview	326
Creating an Alias	328
Exporting an Alias	328
Importing an Alias	329
Creating an AliasLibrary	330
Creating a LibraryBuilder Resource	333

Creating a Design-time Library	336
Using a Resource in the Design-time Library	337
Showing or Hiding Design-time Libraries	338
Managing Resource Conflicts in Design-time Libraries	339
Appendix A Working with a Revision Control System	341
Overview	342
Icons Used by RCS Projects	343
Deleting RCS Projects	343
File Sharing	345
Preparing for File Sharing on Microsoft Windows	345
Preparing for File Sharing on UNIX	345
Using File Sharing	345
Microsoft Visual SourceSafe	348
Visual SourceSafe Setup	348
Using Microsoft Visual SourceSafe	348
Perforce Fast Software Configuration Management System	351
Prerequisites	351
Using Perforce	351
XML Canon	355
Features	355
Prerequisites	356
Checking In and Acquiring Resources	356
Viewing Revision Control Information	362
Deleting XML Canon Projects	363
Tips and Tricks	363
ClearCase	365
Creating or Modifying a ClearCase Project	365
CVS	367
Working with CVS	367
PVCS Version Manager	370
Creating or Modifying a PVCS Project	370
Tips and Tricks for Using Version Control Systems	371
Index	373

Figures

Figure 1	TIBCO Designer startup panel	3
Figure 2	Startup panel Administration options	6
Figure 3	The TIBCO Designer window	8
Figure 4	Project tree in the project panel	10
Figure 5	Global variables in project panel	10
Figure 6	Palette panel changes depending on current selection	12
Figure 7	The three-panel view	14
Figure 8	Three-panel view with palettes selected	15
Figure 9	Layout options	15
Figure 10	Palette mode vs. non-palette mode	16
Figure 11	Adding a resource to a project	29
Figure 12	An example enterprise computing environment	34
Figure 13	A business process flow of an example enterprise	35
Figure 14	A process engine creating process instances	37
Figure 15	A newly created process definition	38
Figure 16	A main process calling a subprocess	44
Figure 17	Developing process definitions	45
Figure 18	Valid transitions	73
Figure 19	An invalid transition	73
Figure 20	The transition dialog	75
Figure 21	Minimizing and maximizing groups	80
Figure 22	The global variable editor	108
Figure 23	Creating a process variable	112
Figure 24	Assigning a value to a user-defined process variable	113
Figure 25	An activity's input tab	118
Figure 26	Mapper check and repair dialog	130
Figure 27	Dragging to the left to change a hint to a statement	133
Figure 28	Examples of mapping required, optional, and nillable elements	135

Figure 29	The XPath formula builder	165
Figure 30	Creating an XPath formula	167
Figure 31	A simple error-handling procedure	175
Figure 32	Propagating errors from a group	179
Figure 33	Propagating errors from a called process	180
Figure 34	The Error Schemas tab	181
Figure 35	The Generate Error Configuration tab	182
Figure 36	Example of process data for error schemas	182
Figure 37	Example of using the Catch activity	183
Figure 38	Example order-entry system	213
Figure 39	Example of periodic processing	213
Figure 40	Example of handling incoming messages	214
Figure 41	Ordering incoming events	224
Figure 42	Multiple event sources to continue a process	225
Figure 43	A typical interaction between web service client and server	231
Figure 44	Service model	232
Figure 45	Relationship between WSDL, Service, Process, and web service clients	234
Figure 46	Input, output, and fault messages of operation must match respective schemas of process	235
Figure 47	Security Policy Association	238
Figure 48	Fault Headers	240
Figure 49	Faults Configured in the Service Resource	240
Figure 50	Advanced Tab for Configured Fault Message	241
Figure 51	SOAPFault Messages	241
Figure 52	Context Resource	242
Figure 53	Mappings on Fault Context	242
Figure 54	Mapping Configured Headers	243
Figure 55	Implementation Process	243
Figure 56	Activities in the SOAP palette	245
Figure 57	MTOM Attachment	249
Figure 58	Advanced SOAP Settings	253
Figure 59	Select a Resource	254
Figure 60	Select Cardinality	255

Figure 61	Partner Link Configuration-Advanced	256
Figure 62	InOut Operations- Advanced SOAP Settings.....	257
Figure 63	Activity Sequence.....	257
Figure 64	Fault Context Data Mapping in Service Resource	258
Figure 65	Fault Context Propagation on Partner Link Configuration	259
Figure 66	Security Context in SOAP Event Source activity Output tab	269
Figure 67	An example of a certificate chain	290
Figure 68	An example of packaged process definitions.....	303
Figure 69	Set BreakPoints dialog.....	313
Figure 70	Setting a breakpoint	314
Figure 71	Process instance stopped at a breakpoint	314
Figure 72	The test panel	315
Figure 73	Multiple process instances in the test panel	317
Figure 74	Save Project Options for Microsoft VSS.....	349
Figure 75	Save Project Options for Perforce Version Control System.....	353
Figure 76	XML Canon tab	357
Figure 77	Browse for an XML Canon category	357
Figure 78	XML Canon Check In Dialog	358
Figure 79	A project checked into XML Canon.....	360
Figure 80	Acquire failed	361

Tables

Table 1	General Typographical Conventions	xxv
Table 2	Startup panel Project options	4
Table 3	Startup panel Administration options	6
Table 4	Example activities	48
Table 5	Activity icon elements	50
Table 6	Activities with Event tabs	55
Table 7	Content types for schema elements	57
Table 8	Icons for schema datatypes	59
Table 9	Process starters	65
Table 10	Group Configuration tab	82
Table 11	Input tab toolbar buttons	121
Table 12	Icons for schema items	124
Table 13	Additional icons for hints	125
Table 14	Datatype validation	127
Table 15	XPath formula builder elements	165
Table 16	Formatting characters in date or time strings	170
Table 17	Example date and time format patterns	172
Table 18	Effects of nesting transaction groups	202
Table 19	JDBC transaction type	204
Table 20	XA transaction type	204
Table 21	JTA UserTransaction type	205
Table 22	Security Contexts Available in ActiveMatrix BusinessWorks	270
Table 23	SSL configuration dialog fields	293
Table 24	Colors in test mode	319
Table 25	Toolbar icons for testing	320
Table 26	Test panel icons	321
Table 27	Revision control system information	362

Preface

TIBCO ActiveMatrix BusinessWorks is a standards-based, easy to deploy solution for companies looking to integrate their enterprise computing environment and automate their business processes. This manual describes how to create process definitions using TIBCO ActiveMatrix BusinessWorks.

Topics

- [Changes from the Previous Release of this Guide, page xxii](#)
- [Related Documentation, page xxiii](#)
- [Typographical Conventions, page xxv](#)
- [TIBCO Product Documentation and Support Services, page xxvii](#)

Changes from the Previous Release of this Guide

There are no changes in this guide during this release.

Related Documentation

This section lists documentation resources you may find useful.

TIBCO ActiveMatrix BusinessWorks Documentation

The following documents form the TIBCO ActiveMatrix BusinessWorks documentation set:

- *TIBCO ActiveMatrix BusinessWorks Concepts* Read this manual before reading any other manual in the documentation set. This manual describes terminology and concepts of TIBCO ActiveMatrix BusinessWorks, and the other manuals in the documentation set assume you are familiar with the information in this manual.
- *TIBCO ActiveMatrix BusinessWorks Getting Started* This manual steps you through a very simple example of designing, deploying, and monitoring a TIBCO ActiveMatrix BusinessWorks process.
- *TIBCO ActiveMatrix BusinessWorks Process Design Guide* This manual describes how to create, edit, and test business processes using TIBCO ActiveMatrix BusinessWorks.
- *TIBCO ActiveMatrix BusinessWorks Palette Reference* This manual describes each of the palettes available in TIBCO ActiveMatrix BusinessWorks.
- *TIBCO ActiveMatrix BusinessWorks Administration* This manual describes how to use TIBCO Administrator to deploy, manage, and monitor TIBCO ActiveMatrix BusinessWorks processes.
- *TIBCO ActiveMatrix BusinessWorks Installation* Read this manual for information on installing one or more components of TIBCO ActiveMatrix BusinessWorks and setting up a TIBCO ActiveMatrix BusinessWorks domain.
- *TIBCO ActiveMatrix BusinessWorks Error Codes* This manual describes errors returned by TIBCO ActiveMatrix BusinessWorks.
- *TIBCO ActiveMatrix BusinessWorks Release Notes* Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.

Other TIBCO Product Documentation

TIBCO ActiveMatrix BusinessWorks is used with other products. You may find it useful to read the documentation for the following TIBCO products:

- TIBCO Designer™ software: TIBCO Designer is an easy to use graphical user interface for design-time configuration of TIBCO applications. TIBCO Designer includes online help for each palette.
- TIBCO Administrator™ software: TIBCO Administrator is the monitoring and managing interface for new-generation TIBCO products such as TIBCO ActiveMatrix BusinessWorks.
- TIBCO Adapter software:
- Third-Party Documentation

Typographical Conventions

The following typographical conventions are used in this manual.

Table 1 General Typographical Conventions

Convention	Use
	TIBCO products are installed into an installation environment. A product installed into an installation environment does not access components in other installation environments. Incompatible products and multiple instances of the same product must be installed into different installation environments. An installation environment consists of the following properties: • Name Identifies the installation environment. This name is referenced in documentation as <i>ENV_NAME</i>. On Microsoft Windows, the name is appended to the name of Windows services created by the installer and is a component of the path to the product shortcut in the Windows Start > All Programs menu.
<i>ENV_NAME</i>	
<i>TIBCO_HOME</i>	• Path The folder into which the product is installed. This folder is referenced in documentation as <i>TIBCO_HOME</i>.
<i>BW_HOME</i>	TIBCO ActiveMatrix BusinessWorks installs into a directory within a <i>TIBCO_HOME</i>. This directory is referenced in documentation as <i>BW_HOME</i>. The default value of <i>BW_HOME</i> depends on the operating system. For example on Windows systems, the default value is C:\tibco\bw\5.12.
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 admin. • In large code samples, to indicate the parts of the sample that are of particular interest. • In command syntax, to indicate the default parameter for a command. For example, if no parameter is specified, MyCommand is enabled: MyCommand [enable disable]

Table 1 General Typographical Conventions (Cont'd)

Convention	Use
<i>italic font</i>	Italic font is used in the following ways: • To indicate a document title. For example: See <i>TIBCO ActiveMatrix BusinessWorks Concepts</i>. • To introduce new terms. For example: A portal page may contain several <i>portlets</i>. 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: <code>MyCommand <i>pathname</i></code>
Key combinations	Key name 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.
	The note icon indicates information that is of special interest or importance, for example, an additional action required only in certain circumstances.
	The tip icon indicates an idea that could be useful, for example, a way to apply the information provided in the current section to achieve a specific result.
	The warning icon indicates the potential for a damaging situation, for example, data loss or corruption if certain steps are taken or not taken.

TIBCO Product Documentation and Support Services

For information about this product, you can read the documentation, contact TIBCO Support, or join TIBCO Community.

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the TIBCO Product Documentation website mainly in the HTML and PDF formats.

The TIBCO Product Documentation website is updated frequently and is more current than any other documentation included with the product. To access the latest documentation, visit <https://docs.tibco.com>.

Documentation for TIBCO ActiveMatrix BusinessWorks™ is available on the [TIBCO ActiveMatrix BusinessWorks™ Product Documentation](#) page.

How to Contact TIBCO Support

You can contact TIBCO Support in the following ways:

- For an overview of TIBCO Support, visit <https://www.tibco.com/services/support>.
- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the TIBCO Support portal at <https://support.tibco.com>.
- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to <https://support.tibco.com>. If you do not have a user name, you can request one by clicking **Register** on the website.

How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the [TIBCO Ideas Portal](#). For a free registration, go to <https://community.tibco.com>.

Chapter 1

Getting Started With TIBCO Designer

TIBCO Designer is an easy to use graphical user interface for creating integration projects.

This chapter and the next give an introduction to TIBCO Designer that is product independent. In this chapter, you learn about TIBCO Designer basics. In the next chapter, you learn about creating and managing projects and working with global variables.

Topics

- [Welcome to TIBCO Designer, page 2](#)
- [Starting TIBCO Designer, page 3](#)
- [TIBCO Designer Administration, page 6](#)
- [TIBCO Designer Interface Overview, page 8](#)
- [Customizing the Display, page 14](#)
- [Accessing Documentation, page 19](#)

Welcome to TIBCO Designer

TIBCO Designer allows you to easily create integration projects for your enterprise computing environment.

TIBCO Designer is available as a graphical user interface to different TIBCO products and is used by those products for configuration. Depending on the product you installed, you can, for example, use TIBCO Designer to create TIBCO ActiveMatrix BusinessWorks process definitions or create or modify adapter configurations.

- TIBCO ActiveMatrix BusinessWorks is a scalable, extensible, and easy to use integration platform that allows you to develop, deploy, and run integration projects. TIBCO ActiveMatrix BusinessWorks also includes an engine that executes the process, and a web-based GUI for monitoring and managing run-time components.
- Adapters allow you to configure the interface between an external system, such as an SAP R/3 application or a database, and the TIBCO ActiveEnterprise environment. Adapters are available as separate products.
- Custom adapters are created using the TIBCO Adapter SDK. You can prepare an adapter configuration for custom adapters using the Adapter Resources and Adapter Schemas palettes, which are discussed in *TIBCO Designer Palette Reference*.
- TIBCO ActiveMatrix BusinessWorks Collaborator gives companies the ability to coordinate business activities, measure their efficiency, and optimize them over time. The product facilitates complete visibility into business activities, along with the ability to collaborate on the modeling and modification of the rules and flows that define business those activities. TIBCO ActiveMatrix BusinessWorks Collaborator uses TIBCO Designer for configuration of FormFlows processes and for preparing Enterprise Archive files.

Starting TIBCO Designer

The following sections describe how to start TIBCO Designer and explain the options available once TIBCO Designer starts.

To Start TIBCO Designer

- Under Microsoft Windows:
Choose **Start > All Programs > TIBCO > TIBCO Designer *n.n* > Designer *n.n***
or
Invoke *install-path\tibco\designer\i.n.n\bin\designer*
- Under UNIX:
Invoke *install-path/tibco/designer/i.n.n/bin/designer*

Startup Options

When you launch TIBCO Designer, the startup panel is displayed:

Figure 1 TIBCO Designer startup panel

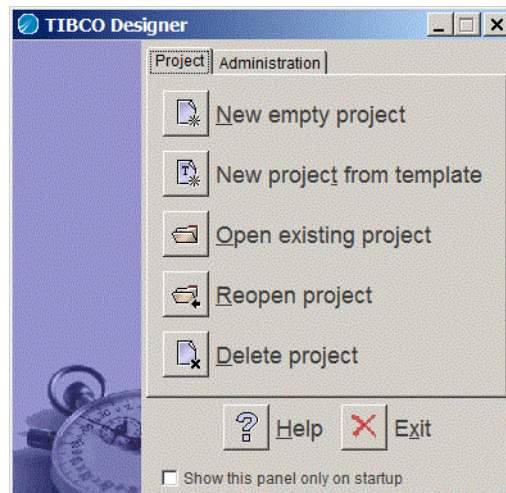


Table 2 describes the startup options. Table 3, [Startup panel Administration options](#), describes the options available when you select the Administration tab.

Table 2 Startup panel Project options

Option	Description
New empty project	Opens a new empty project in TIBCO Designer. An empty project includes the TIBCO Designer default palettes and their resources. By default when you open a new project, TIBCO Designer prompts you immediately where you wish to save it. You may provide the location or click the Cancel button. If you do not want to see this dialog each time you create a new project, choose Edit > Preferences > General and unselect Show save dialog for new project.
New project from template	Opens a predefined project template. A project template is a pre-built project. It can contain folders, configured resources, and partially configured resources. Resources are the components of each project (see Resources on page 10). A project template can be preconfigured to include all the resources you may need for a certain type of project (for example, Web Services configuration). Using a template makes it possible to leverage an existing configuration when creating new projects. See Saving Projects on page 26 for information on creating project templates.
Open existing project	Opens an existing project. See Opening and Reopening Projects on page 28.
Reopen project	Allows you to choose from a list of recently saved projects. TIBCO Designer may prompt for information, for example, a password.
Delete project	Allows you to delete a project. See Deleting Projects on page 31.

Table 2 Startup panel Project options

Option	Description
Help	Displays TIBCO Designer documentation. You may be prompted for your browser location if you are using TIBCO Designer for the first time. Information about browser locations on some operating systems is included in the prompt screen. You need to specify this path only once. After that, TIBCO Designer remembers the location even if you uninstall the current version and install a new version.
Exit	Exits TIBCO Designer.
Show this panel only on startup	If checked, the startup panel is only displayed during startup and closed after you've made your selection. If cleared, this panel reappears when no other TIBCO Designer windows are open. Leaving the panel on screen can be useful for project maintenance.

TIBCO Designer Administration

When you start TIBCO Designer, the startup panel allows you to open projects (see [Startup Options on page 3](#)). You can also use the startup panel to perform some TIBCO Designer administrative tasks. When you choose the Administration tab, the panel presents the choices shown in [Figure 2](#):

Figure 2 Startup panel Administration options

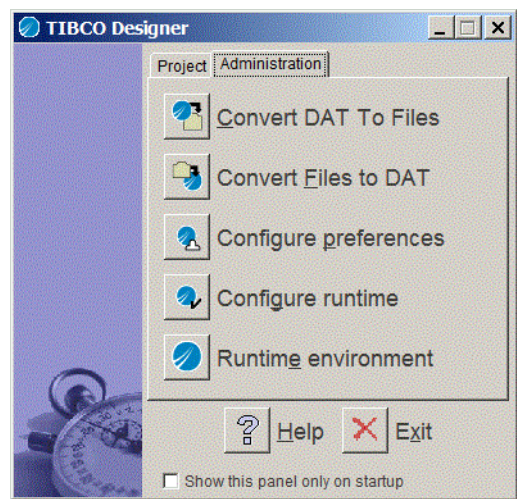


Table 3 Startup panel Administration options

Option	Description
Convert DAT to files	Displays a dialog that lets you specify the .dat file you wish to convert and the folder for the multi-file project. Because .dat files are a legacy format, you must convert to a multi-file project before you can open a project from TIBCO Designer.

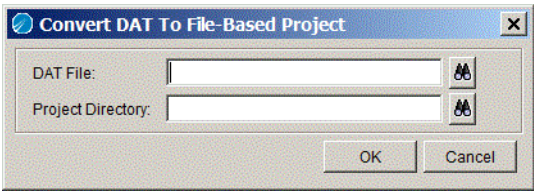


Table 3 Startup panel Administration options

Option	Description
Convert files to DAT	Displays a dialog that lets you choose a project directory and a .dat file. When you specify both, TIBCO Designer converts the multi-file project in the directory to the .dat file.
Configure preferences	Displays the Preferences dialog. See <i>TIBCO Designer User's Guide</i> for more information.
Configure runtime	Allows you to configure the TIBCO Designer runtime environment. You have these options: - Extended Class Path—classpath to be used by TIBCO Designer. Users can specify file names or directories. If they specify directories then all .class, .zip and .jar files will be loaded. The order in which they are loaded depends on the file system. - Palette Path—Location from which TIBCO Designer loads palettes. - Maximum Heap Size—Maximum JVM heap size. - User Directory—Default location for the application to store files. - Command Line Arguments—Allows command line arguments to be passed to Designer. Currently -d (debug) is supported. If you specify -d, the log that is sent to the Console becomes more detailed.
Runtime environment	Displays TIBCO Designer runtime information. This information, which includes the palette name and version information, supporting product name and version information, and Java property and value information, can be useful for debugging or during interaction with Technical Support. Use the Export Runtime Settings button to create a file with all pertinent information.

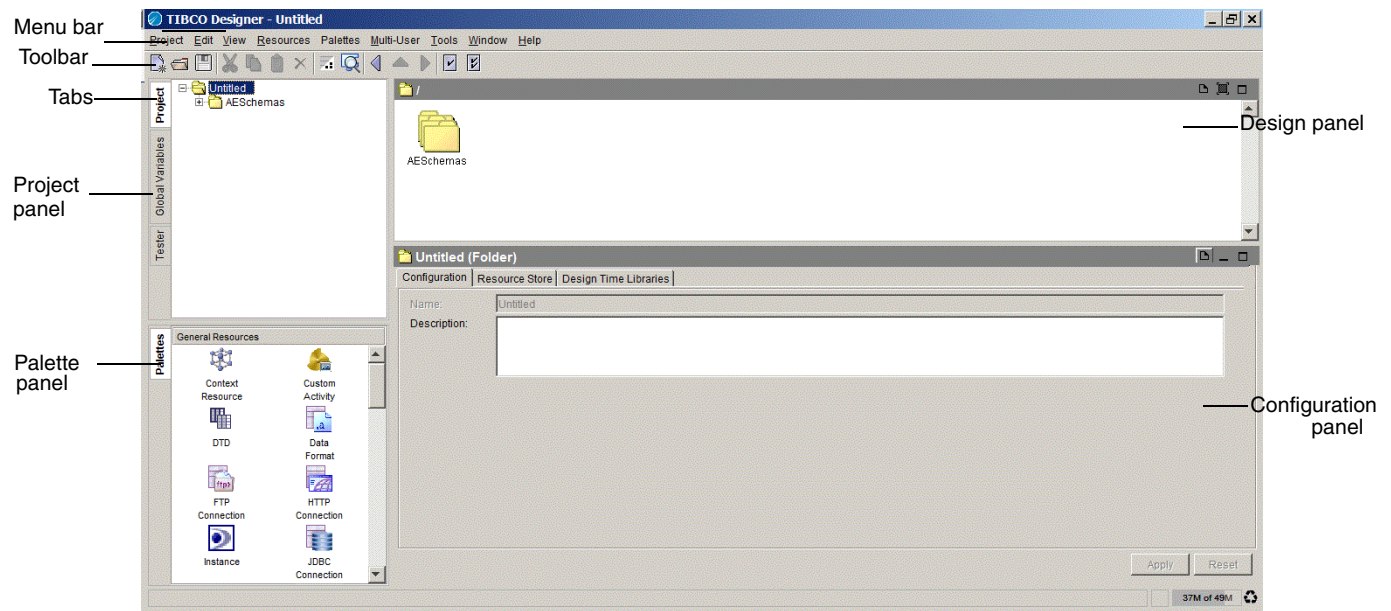
TIBCO Designer Interface Overview

The TIBCO Designer interface allows you to perform various functions. This section describes the TIBCO Designer main window and explains what you see in each of its panels.

Main Window

Figure 3 illustrates the TIBCO Designer window.

Figure 3 The TIBCO Designer window



The TIBCO Designer window has these components:

- Menu bar and Menus. See *TIBCO Designer User's Guide*.
- Toolbar icons. See *TIBCO Designer User's Guide*.
- Tabs in the leftmost area allow you to change what is displayed in the panel. See [Customizing the Display on page 14](#).



When something in the design panel or the configuration panel is selected, the panel is highlighted. This helps you see at one glance where the focus is.

- Four panels, which are (starting in the top left corner and continuing clockwise):
 - Project panel (can display the project tree or the project's global variables)
 - Design panel
 - Configuration panel
 - Palette panel



You can rearrange the panels and what they display. For example, the project panel and palettes can be combined to share one set of tabs. See [Customizing the Display on page 14](#) for more information.

The following sections explain the contents of each panel.

Project Panel

A *project* contains resources that implement the enterprise integration. This includes services (producers and consumers of information), any business logic that may be applied to that information, and deployment information.

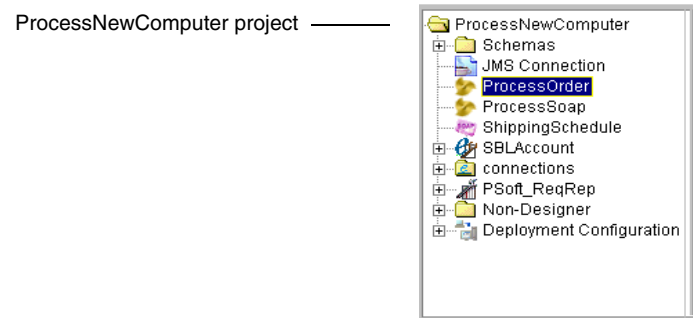
By default, the project panel allows you to view the [Project Display](#) or [Global Variables Display](#). When used in conjunction with other projects, the panel may be used for other purposes, for example, in conjunction with the TIBCO ActiveMatrix BusinessWorks tester.

Project Display

With the `Project` tab selected, the project panel displays the project tree. This includes the top-level (root) folder and the hierarchy of resources. The hierarchy of folders and resources corresponds to the hierarchy of folders and files in the project folder.

[Figure 4](#) illustrates an example project, `ProcessNewComputer`, in the project panel. Multiple TIBCO products were used to create the integration project: it contains two TIBCO ActiveMatrix BusinessWorks process definitions (`ProcessOrder` and `ProcessSoap`) and a Siebel adapter (`SBLAccount`).

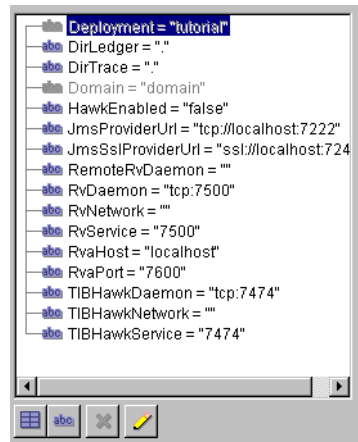
Figure 4 Project tree in the project panel



Global Variables Display

Global variables are associated with each project. To display them, click the Global Variables tab of the project panel. Clicking the pencil icon displays the global variable editor. See [Global Variables on page 107](#) for more information.

Figure 5 Global variables in project panel



Resources

Resources are the components of a project. A simple TIBCO Designer resource corresponds to an object in a TIBCO application, such as an adapter configuration, an adapter service, a process definition, or an FTP activity.

Resources can be complex and contain other resources, much like a folder can contain other folders on your computer's file system. For example, an adapter configuration may contain multiple folders with multiple publisher or subscriber service resources.

Each top-level resource (for example, each adapter configuration) corresponds to a file in the project's hierarchy of files in the project directory. This design allows developers to use a source control system and to check out only the top-level resources they are working with from a source control system, thus sharing their work.

Most resources have context-sensitive help available for the configuration of that resource. Right-click on the resource and choose **What Is This?** from the popup menu for more information on configuring the resource.



If TIBCO Designer cannot determine the type of a resource, it displays it as a special icon. This can mean, for example, that the palette for that resource is not installed, that the file is not really part of your project, or that it has a misleading extension.

For example, source control systems may hide files from the user. If you open a Designer project that was under source control without the revision control system, these files will become visible and treated as unknown resources.

Palette Panel

Palettes organize resources and allow you to add them to your project. You select resources in the palette panel and drag and drop them into the design panel to add them to your project.

TIBCO Designer contains a small number of native palettes. In addition, each TIBCO application you install adds one or more palettes during installation.

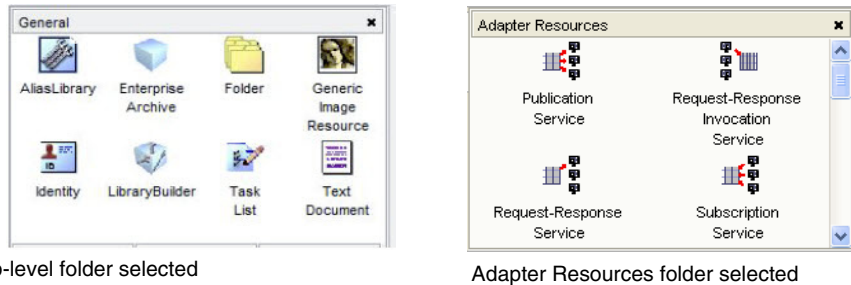
Which palettes are displayed depends on:

- the installed TIBCO products
- the resource selected in the project tree
- your preferences (see [Customizing the Display on page 14](#))

Current Selection and Palette Panel Display

When the default view is set as your view preference, the current selection in the project tree determines which palettes are displayed in the palette panel.

Figure 6 Palette panel changes depending on current selection



For example:

- Select the top-level project folder to see a palette for each adapter and some other palettes for general resources.
- Select the Adapter Services folder of an adapter in the project tree to see a palette of service resources. Drag any service resource into the design panel to add that resource to that adapter.

You can change your view preferences to change what's displayed in the palette panel. See [Customizing the Display on page 14](#) for more information about how TIBCO Designer functions in palette mode.

Design Panel

The design panel displays the current resource selected in the project panel. For resources that contain other resources, the contents of the selected resource are shown in the design panel. For example, if you select a folder, its contents is displayed.

Configuration Panel

The configuration panel allows you to specify various configuration options for each resource. The type and the purpose of the selected resource determine the contents of the configuration panel. Usually there are one or more tabs in the configuration panel that allow you to access the various configuration options. The tabs organize the configuration options for the resource.

Click the help icon  in the top right corner of the configuration panel for online help on the current selection.

After you have added the configuration information, you must click the **Apply** button for each tab. If you decide you do not want to add the configuration information, click **Reset** before you apply any changes to return to the previous values for each field in the tab.

Customizing the Display

You can customize how TIBCO Designer displays panels and palettes. This section gives an overview of the most frequently used display preferences.

Display preferences and other preferences are saved when you exit TIBCO Designer, even if you do not save your project. Display preferences are maintained for each user, even if that user completely uninstalls the product and installs a different version.

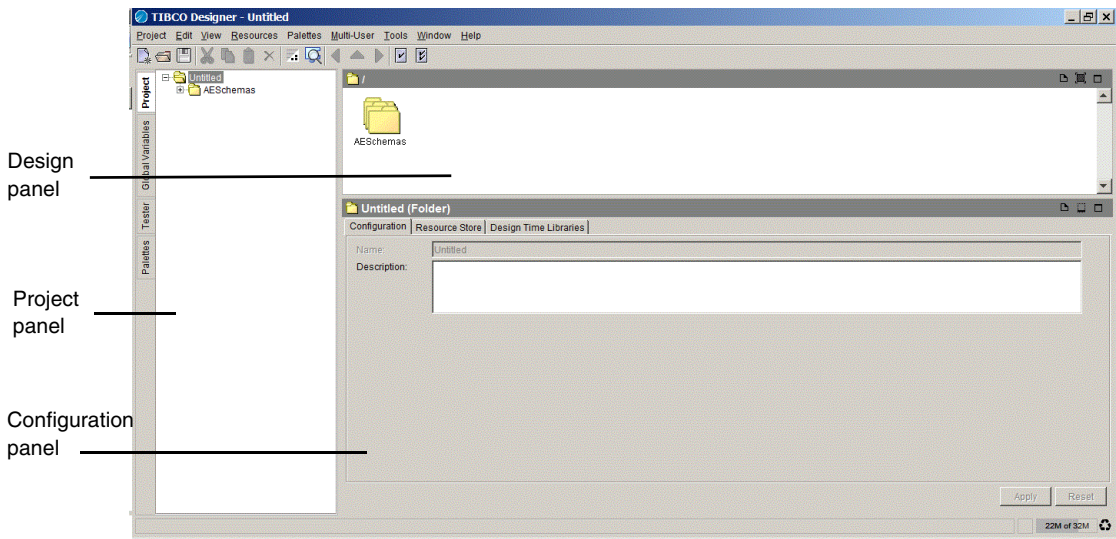


To return to the default settings, select **Edit > Preferences** and click **Restore Default Settings**.

Choosing Panel Layout

If you prefer to view either the project tree or the palette panel, but not both in the left panel, you can set TIBCO Designer to display the three-panel view shown in [Figure 7](#). The view also uses drop downs, rather than tabs to access the project, global variables or palettes.

Figure 7 The three-panel view



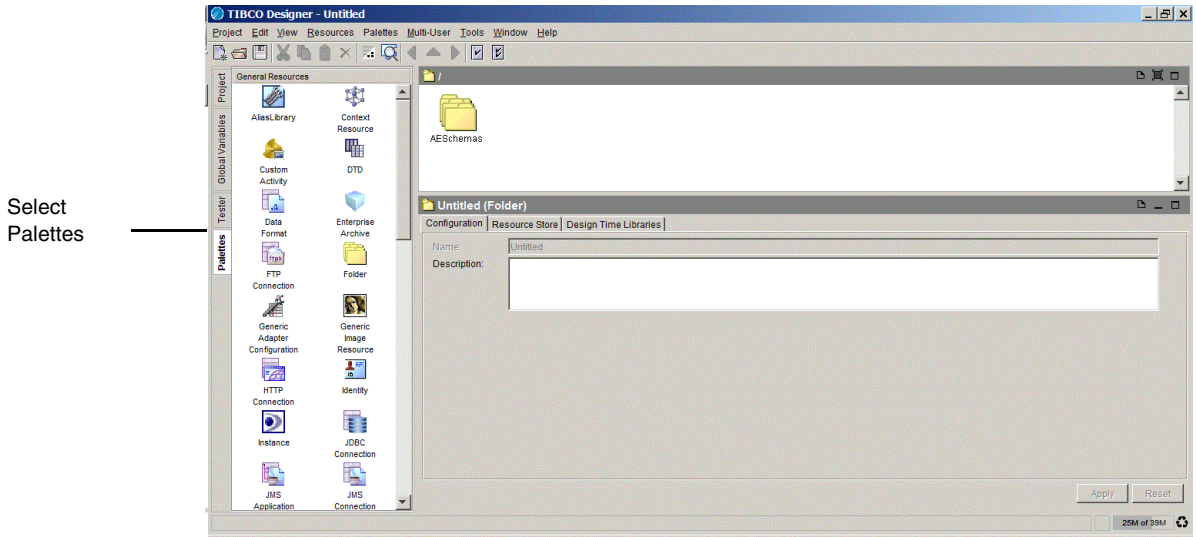
To Choose the Three-Panel View

1. Choose **Edit>Preferences>View**.

2. Under **Layout**, select the appropriate icon and click **OK**.

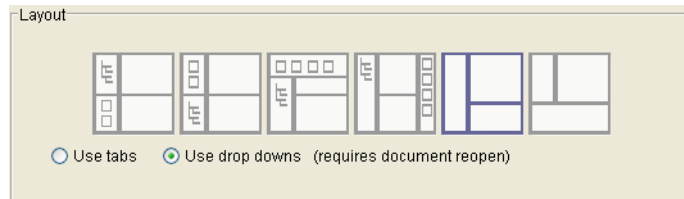
To navigate to palettes in this view, click the **Palettes** tab on the left (see [Figure 8](#)). The next diagram shows the results of this action.

Figure 8 Three-panel view with palettes selected



Additional layout options are available when you choose **Edit > Preferences > View**. If you select the **Use tabs** or **Use drop downs** option, you must close and reopen your project to make the change visible.

Figure 9 Layout options



Choosing Palette Mode or Non-palette Mode

TIBCO Designer allows you to change the palette panel display to use palette mode or non-palette mode.

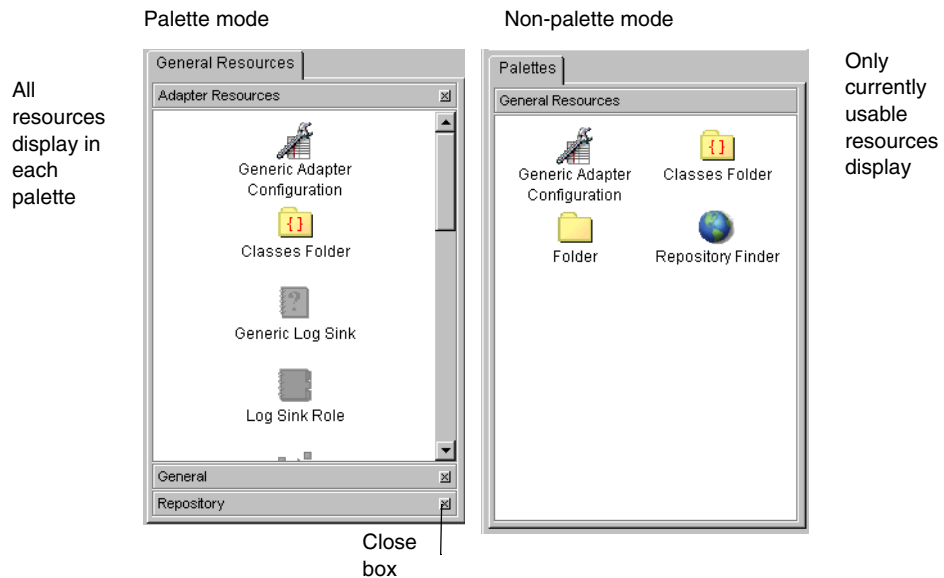
- In palette mode, each resource is shown in the palette it belongs to, and each palette shows all resources. In that case, unusable resources are grayed out.
- In non-palette mode, resources are displayed directly in the panel, and only currently usable resources are displayed.

While in palette mode, you can close individual palettes using the close marker (X) on the right. If you don't see close markers, choose **Palettes > Options > Show Close Boxes**. To hide close markers, choose **Palettes > Options > Hide Close Boxes**.

To redisplay a closed palette, choose **Palettes > Browse**, then locate and select the palette.

Figure 10 illustrates the palette panel in palette mode and non-palette mode.

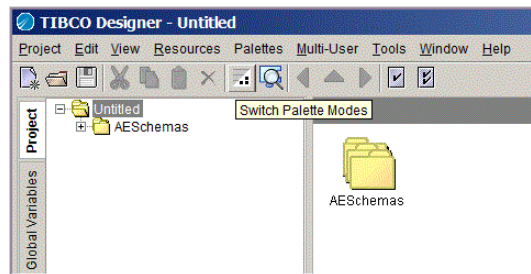
Figure 10 Palette mode vs. non-palette mode



To Switch Palette Modes

- Choose **Palettes > Options > Switch Palette Modes**
- or

- Click the **Switch Palette Modes** button located in the tool bar.



Displaying Palettes in a Separate Window

You can display the palette panel in a separate window by choosing **Palettes > Options > Show Palettes in New Window**. You can also access this menu command from the right-button menu of any palette in the palette panel.

To restore the palette panel to its location in the main window, close the separate window in which the palette panel is displayed.

Working With User Palettes

User palettes allow you to save a collection of resources to a predefined location and either share it with other users or use it yourself at a later time.

To Create a User Palette

To add a user palette, follow these steps:

1. Choose **Palettes > My Palettes > New Palette**.
2. Specify the name of the palette.
When you later save your project, the palette is saved to the location specified by the `User Palette Directory` General Preference.
3. Drag resources from the project tree or from the design panel into the user palette.
4. When you save your project, the custom palette is then saved to the location specified by the `User palette directory` under the `General` tab in the Preferences dialog.

To Load a User Palette

1. Choose **Palettes > My Palettes > Reload Palettes**.

To Delete a User Palette

1. Make sure the palette is loaded.
2. Choose **Palettes > My Palettes > Delete Palettes**.

You are prompted for the palette(s) you wish to delete.

Accessing Documentation

Documentation for TIBCO ActiveMatrix BusinessWorks is available in several ways:

- If you are using Microsoft Windows, you can use the **Start** menu to access TIBCO Designer documentation.
- From TIBCO Designer:
 - Click **Help > Designer Help** at any time to view this manual, which discusses how to use TIBCO ActiveMatrix BusinessWorks regardless of the application you are building.
 - Use **Help > Help For** to access product-specific documentation.
 - Right click on most resources and choose the **What Is This?** menu item to view specific help for that resource. If information is displayed in the Configuration panel, you can also click the Help icon  for online help.



When you invoke Designer help for the first time, you are prompted for the location of your web browser. You only need to supply this location once. Location information is included in the prompt.

Chapter 2

Managing Projects and Resources

Projects are the key organizational principle for the configuration information you specify with TIBCO ActiveMatrix BusinessWorks.

This chapter explains how to manage projects and the resources inside them.

Topics

- [Overview of Projects, page 22](#)
- [Creating Projects, page 24](#)
- [Validating Projects, page 25](#)
- [Saving Projects, page 26](#)
- [Opening and Reopening Projects, page 28](#)
- [Adding Resources To Your Project, page 29](#)
- [Deleting Projects, page 31](#)
- [Tips and Tricks for Working With Projects, page 32](#)

Overview of Projects

A *project* is a collection of resources, including, for example, adapter resources and process definitions. Together, these resources define the configuration of your integration project. In the TIBCO ActiveMatrix BusinessWorks window, a project is represented by the top-level (root) folder in the project panel. The top-level folder is initially named `Untitled` and is renamed to the name of the project when you save the project for the first time.

Each TIBCO Designer window contains only one project. If you open a second project, TIBCO Designer opens a new window.

When you close a project, the startup panel remains available for project maintenance or for opening a different project unless you selected the **Show this panel only on startup** check box on that window.

Project Structure

When you save a project, TIBCO Designer creates a hierarchy of folders and files in the location you choose for the project.



When you supply a project location, be sure no other files reside in that folder. TIBCO Designer removes any existing files before placing the project files into the folder.

When you create a multi-file project:

- There is one file per top-level resource. For TIBCO ActiveMatrix BusinessWorks, that may mean one file per process. For adapters, more resources may be considered part of a top-level resource.
- The project root directory identifies the project. The project root can be located anywhere in the file system and is determined when you first save the project. All components of a project are located under this common project root.
- Directories in the file system become folders in TIBCO Designer. However, not all folders in TIBCO Designer are directories in the file system:



- Folders created from a Folder resource in the General Palette (displays a multi-folder icon) become folders in the file system.



- Other folders, such as the Adapter Services folder inside an adapter configuration are logical folders. These folders only exist in memory in the resource that holds them. The actual data for these elements is stored in the file of the top-level resource. For example, the Adapter Services data are stored in the Adapter Configuration file.

- TIBCO Designer creates a file named `vcrepo.dat` in the project root directory when you first save the project. This file is used to store properties such as display name, TIBCO Rendezvous encoding, and description. This file can be used for identification in place of the project root directory and can be used as the repository locator string (`repoUrl`).

Using a Version Control System

Multi-file projects support the use of different version control systems because they consist of separate files for each versionable component.

Once the root directory is defined under the control of a version control system, standard version control system tools can be used. The following version control systems are supported and discussed separately:

- [File Sharing on page 345](#)
- [Microsoft Visual SourceSafe on page 348](#)
- [Perforce Fast Software Configuration Management System on page 351](#)
- [XML Canon on page 355](#)
- [ClearCase on page 365](#)
- [CVS on page 367](#)
- [PVCS Version Manager on page 370](#)

Project Templates

A project template is a pre-built project. It can contain folders for organization, configured resources, and partially configured resources. You can use a project template as the foundation for other projects similar in nature. Using a template, you can leverage your work when performing similar configurations.

Creating Projects

You create a new project using the startup panel when starting TIBCO ActiveMatrix BusinessWorks. You can also choose **Project > New Project** from the TIBCO Designer menu bar with TIBCO Designer already open. In that case, TIBCO Designer opens a new window for the new project.

TIBCO ActiveMatrix BusinessWorks allows you to create a project from scratch or to create a template-based project.

- **New Empty Project** — A new project contains a single AESchemas folder that will be used for adapter schema resources. See [Project Structure on page 22](#) for more information.
- **New Project from Template** — When you save a project as a template (**Project > Save As Template**), you can later load that template and customize it to create a new project. See [Project Templates on page 23](#).

When you create a new project, you are, by default, prompted immediately to save it. See [Saving Projects on page 26](#) for a discussion of the information you must supply.

Validating Projects

After you have created a project, you add resources to it and supply configuration information for your resources.

Before you prepare a project for deployment, it is critical that you validate it. TIBCO Designer includes reference-checking and other validation facilities that allow you to make sure a project is internally consistent. This is essential if you intend to run the project, or hand it to another user.

During validation, each resource always checks for broken references. Many resources have other resource-specific validation behavior.

To Validate a Resource

- Select the resource to be validated, then choose **Resources > Validate Resource** from the menu bar.
- With the resource selected, choose the Validate Resource icon. 

To Validate All Resources

- Choose **Project > Validate Project for deployment**.
- Click the Validate Project for Deployment icon. 

Note that TIBCO Designer handles references as strings. TIBCO Designer will help keep these references up to date, for example, when you move a resource to a different location. It is, however, possible to have "broken" references, for example, if you delete a resource and ignore the warnings displayed by TIBCO Designer. You can use the validation commands to find broken references.



By default, TIBCO Designer prompts whether you wish to perform reference checking each time you perform an activity that might result in a broken reference (move, rename, and so forth). You can change the default behavior using the **Edit > Preferences > References** tab.

Saving Projects

When you save a project, you can save it as a multi-file project, or under XML Canon. This section explains how to save a multi-file project. See [XML Canon on page 355](#) for information about using XML Canon.

When you save a multi-file project, you have these choices:

- **Project Directory** — The directory that will contain the project files. Click **Browse** to select the directory.



Designer will remove any files in this directory when you save the project.

- **TIBCO Message Encoding** — Character Encoding used for the communication between TIBCO product components in this project at design time (debug mode), or if the project is running as a legacy local file-based project. The communication transport could be either TIBCO Rendezvous or TIBCO Enterprise Message Service. You have two choices:
 - **ISO8859-1 (Latin-1)**—Preferred encoding for projects that deal only with English and other Western European languages that belong to the ISO Latin-1 character set. If this encoding is used for languages that do not belong to the Latin-1 character set (such as Japanese, Arabic, etc.), data loss may result.
 - **UTF-8**—Preferred encoding for projects dealing with languages not belonging to the Latin-1 character set. This includes most languages except for English and other western European languages.



After deployment, the encoding setting of the TIBCO Administration Server will supersede this encoding. See the *TIBCO Administrator Server Configuration Guide* for more information.

- **Multi-User System** — Allows you to use a multi-user system such as file sharing, Perforce, or Visual SourceSafe. See [Appendix A, Working with a Revision Control System, on page 341](#) for more information.

After you have saved a project to a repository, you can select the project's **Project Settings** tab to:

- View information about the project. The information displayed depends on how the project was saved.
- View and change the project's messaging encoding for the data communication among the components in this project. This is only used in

design mode, or when the project is running as a legacy local file-based project.

To Save a Project

1. In the main window, do one of the following:
 - Choose **Project > Save**.
 - Choose **Project > Save As** and specify the storage directory.
 - Click the **Save** icon .
2. In the dialog that is displayed, make sure the **Multi-File Project** tab is selected and provide the following information:
3. Click **OK**.

To Save a Project as a Template

1. Choose **Project > Save As Template**.
2. Provide the appropriate information, which is the same as discussed in [Saving Projects on page 26](#).

Opening and Reopening Projects

You can open a project in two ways:

- From the startup panel when you launch TIBCO ActiveMatrix BusinessWorks.
- Choose **Project > Open** from the TIBCO ActiveMatrix BusinessWorks main window if Designer is already open. In that case, TIBCO Designer will create a new window for your project.



If you need to open a .dat project, you must convert it first:

1. In the startup panel, click the **Administration** tab.
2. Choose **DAT to Files**.
3. Supply the name of the project directory when prompted.

You can then open the multi-file project from TIBCO Designer.

If you are opening a project under a revision control system, you need to provide the appropriate information. See [Appendix A, Working with a Revision Control System](#).

You can reopen a project you opened recently in two ways:

- From the startup panel when you launch TIBCO ActiveMatrix BusinessWorks.
- Choose **Project > Reopen** from the TIBCO ActiveMatrix BusinessWorks main window if Designer is already open. In that case, TIBCO Designer will create a new window for your project

Adding Resources To Your Project

Once you have created or opened a project, you can add resources to your project. To add a resource, you first select it in the palette panel, then drag and drop it into the design panel.

To Add a Resource to Your Project

1. Select the palette in which the resource can be found. For example, you find an adapter configuration resource in the palette named after the adapter.



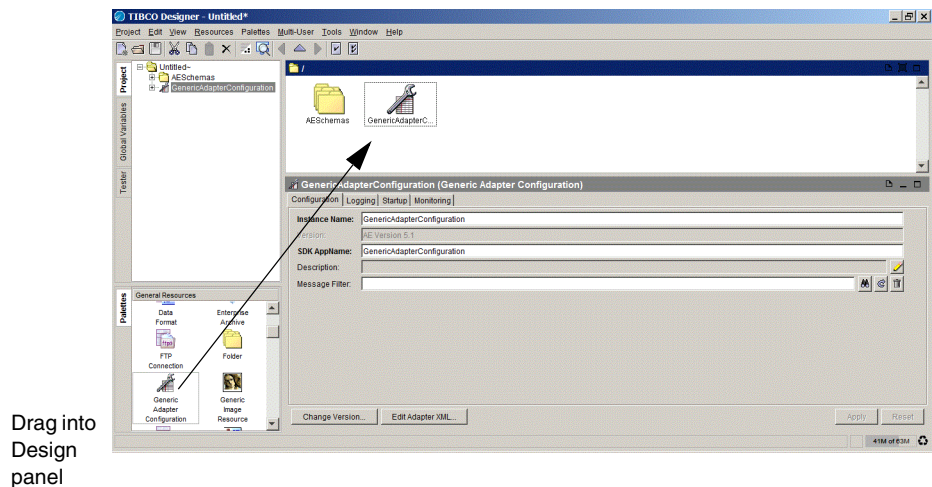
If the palettes are not visible in your palette panel, click the Switch Palette Modes icon.

2. Select a resource within the palette and drag and release it in the design panel.

The resource instance is displayed in the design panel and also added to your project tree. The configuration panel allows you to specify configuration information for the resource.

Figure 11 illustrates adding a resource to a project.

Figure 11 Adding a resource to a project



Drag into
Design
panel



You can also add resources in other ways:

- In the palette panel, select the resource and choose **Add This To The Project** from the right-button menu.
- In the design panel, right click on an empty area (not on a resource) and select **Add Resource** from the right-button menu, then choose the appropriate submenu.
- You can enable resources to be added using double-clicks. Go to **Edit > Preferences** and select the **View** tab. Under **Palettes**, select **Initiate drag-n-drop through double-clicks**.

If Adding a Resource Results in an Error

Whether adding a resource is possible depends on what is currently displayed in the design panel.

If you try to add a resource that cannot be added to the current resource, an error results.

For example, if the root folder is displayed in the design panel, you can add an adapter instance. If any other resource is displayed, you cannot add the adapter instance.



Ideally, all resources that cannot be dragged into the design panel should be grayed out (palette mode) or not visible (non-palette mode). For some custom palettes that may not always be true.

Deleting Projects

You do not delete projects from the TIBCO Designer main window but from the startup panel.

To Access the Startup Panel

- If TIBCO Designer isn't running, start it.
- If TIBCO Designer is running, close all windows. The startup panel is displayed unless `Show this panel only on startup` has been checked.

To Delete a Project

1. In the startup panel, make sure the Project tab is selected.
2. Click **Delete Project**.
3. In the panel that appears:
 - a. Specify the project directory
 - b. Specify a version control system if the project was used in conjunction with a version control system.
 - For File Sharing, any user with access to a project can delete the project.
 - For other version control systems you must make sure that both the (local) project directory and the directory you specify for the version control system are correct. You must also be sure to specify a user that has appropriate privileges for deleting the project.



You cannot delete projects based on XML Canon from TIBCO Designer. To delete such a project, you must use a WebDAV client.

Tips and Tricks for Working With Projects

This section contains additional information on using multi-file projects.

- **Use ASCII project names.** You must use an ASCII name for the project when saving the project from TIBCO Designer. Project names must not use the following characters: | / \ " ' : ?.
- **Avoid naming collision.** You cannot place a multi-file project and a single-file (.dat) project into the same directory.
- **Place schema in the AESchemas folder.** If you edit your project file in an XML editor and define schema outside the /AESchemas folder, the schema are placed in a directory called __NON__DEFAULT__SCHEMA__FOLDER__ in /tibco/public/<type> where type is the kind of object (that is, class, scalar, union, and so forth).

It is cleaner to have schema files under /AESchemas. In addition, it is required you place schema files into /AESchemas if you wish to edit your project using TIBCO Designer.

Note that while editing schema files is not prohibited, you do so at your own risk.

- **Consider using global variable groups.** Use global variable groups to allow multiple developers to work on global variables simultaneously. Each group has its own file in the multi-file project. See [Global Variables on page 107](#) for more information.

Note, however, that an excessive amount of global variables (over 500) can lead to problems.

Chapter 3 Processes

TIBCO ActiveMatrix BusinessWorks allows you to graphically model your business processes and execute them automatically. This chapter describes how to create business processes in TIBCO ActiveMatrix BusinessWorks.

Topics

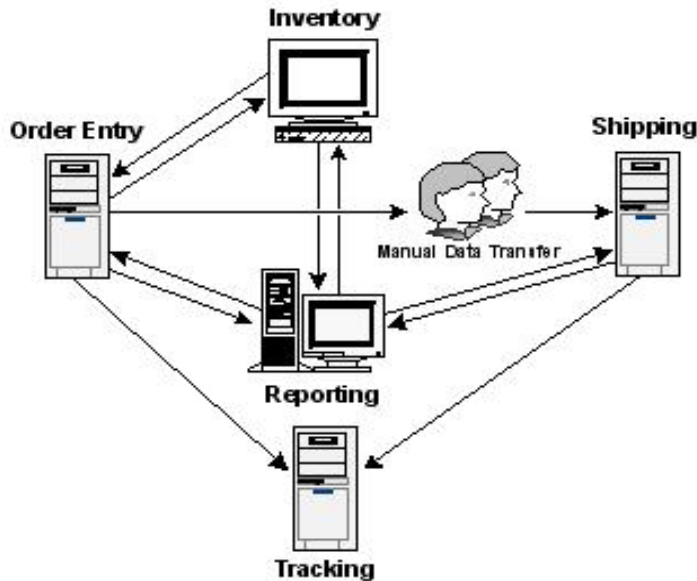
- [Business Process Modeling, page 34](#)
- [Overview of Processes, page 37](#)
- [Process Definitions, page 38](#)
- [Activities, page 39](#)
- [Transitions, page 40](#)
- [Groups, page 41](#)
- [Shared Configuration Resources, page 42](#)
- [Subprocesses, page 43](#)
- [Developing Process Definitions, page 45](#)

Business Process Modeling

Most businesses choose the best application or environment for processing each component of their core business. In a typical enterprise, the flow of information and processing between each application is what drives the day-to-day operations of the business. For example, the order-entry application receives and processes orders based on availability information taken from the inventory system. The tracking system relies on order information and shipping information. Also, accurate and up-to-date reporting information is required from all systems.

Figure 12 illustrates an example enterprise computing environment with various systems running in different environments.

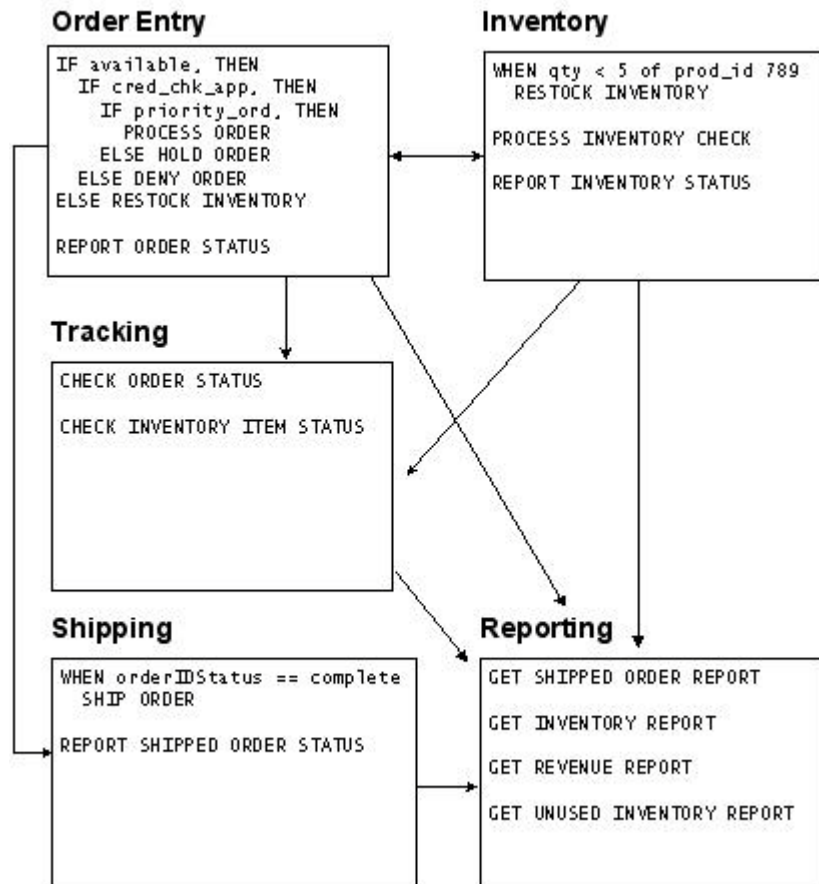
Figure 12 An example enterprise computing environment



Many companies implement the business rules that tie the systems together using custom-written code or by manual processes. Tying together different systems from different vendors that run in different environments is a labor-intensive and error-prone process that usually takes months of planning and implementation. Also, because the task of creating the custom business logic is so complex, businesses often rely on manual, paper-based processes instead of automating the process for greater efficiency.

Figure 13 illustrates a business process flow, sometimes known as a workflow, that describes the business rules between the various systems in an enterprise.

Figure 13 A business process flow of an example enterprise



The business process flow describes an integrated enterprise that contains order-entry, inventory, shipping, tracking, and reporting systems. Each of these systems have rules for processing incoming data and more rules for passing data between the systems. These rules are characterized by business processes, such as the REPORT SHIPPED ORDER STATUS process in the shipping system.

These business rules in themselves can involve complex processing and automating these processes is crucial to lowering the total cost of operating the complete enterprise environment.

TIBCO ActiveMatrix BusinessWorks Solves Enterprise Integration Problems

An ideal solution for handling business process automation would be a tool that can handle the different environments and applications and allow you to create programmatic business rules easily. That tool should also allow you to automate your business processes for the greatest efficiency.

TIBCO ActiveMatrix BusinessWorks allows you to model business processes with a graphical tool. You can use the TIBCO ActiveMatrix BusinessWorks process definition palette to diagram complex business logic easily. Once the business rules have been specified, TIBCO ActiveMatrix BusinessWorks can execute the business processes, allowing you to easily automate the critical functions of your business.

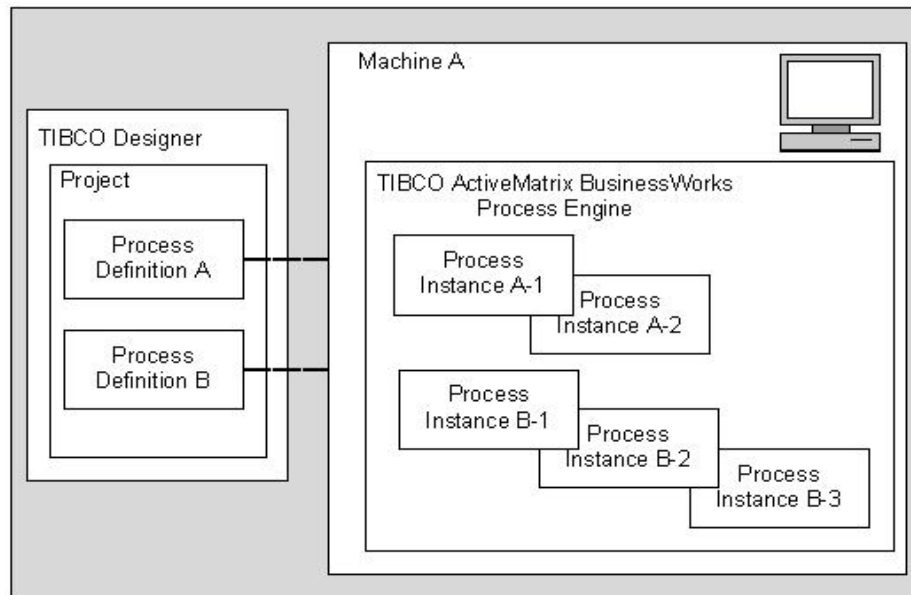
TIBCO ActiveMatrix BusinessWorks can help you specify the business logic and automate the processing of the interaction between the systems in your enterprise. This allows you to reduce the time to implement an integrated, enterprise-wide computing environment and ultimately lower the cost of deploying and maintaining the system.

Overview of Processes

A process definition is the graphical representation of your business process. You develop and test process definitions using TIBCO Designer. The process definition is executed by a TIBCO ActiveMatrix BusinessWorks *process engine*. A process engine creates instances of process definitions. These *process instances* automate your business processes by executing the business process described by the process definition.

Figure 14 illustrates the relationship between process definitions, a process engine, and process instances.

Figure 14 A process engine creating process instances



Process engines are started using TIBCO Administrator after you deploy your project. For more information about deploying a project, see *TIBCO ActiveMatrix BusinessWorks Administration*.

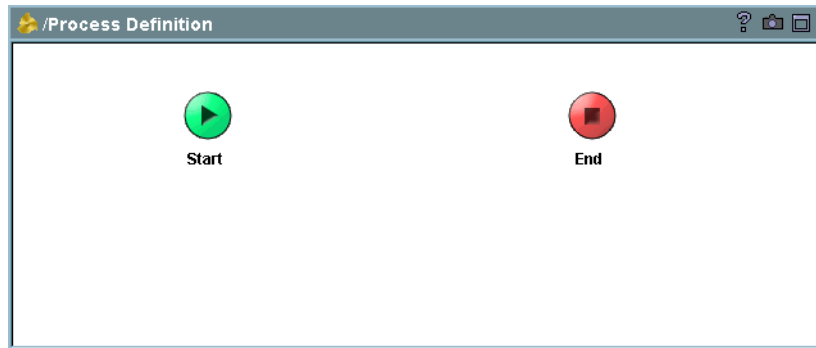
The remainder of this manual describes how to create the process definitions that eventually become running process instances.

Process Definitions

A *process definition* is a graphical representation of your business process model. You create process definitions by dragging and dropping a Process Definition resource from the Process palette to the design panel.

Selecting the process definition in the project panel changes the palette panel and the design panel to allow you to create your business process model. The palette panel contains a set of activity palettes for use in the process definition. The design panel displays the business process model. Newly created process definitions contain a Start activity and an End activity. [Figure 15](#) illustrates a newly created process definition.

Figure 15 A newly created process definition



The process definition describes the business process. When the process definition is executed, it is known as a process instance.

Process definitions consist of these components:

- [Activities](#)
- [Transitions](#)
- [Groups](#)
- [Shared Configuration Resources](#)
- [Subprocesses](#)

The following sections describe these components. See [Developing Process Definitions on page 45](#) for a description of how to develop your process definitions.

Activities

Activities are the individual units of work in a process definition. Activities are generally operations that interface to external systems, but activities can also perform internal processing. Activities are available on the various palettes in TIBCO Designer. Each palette has a set of activities that can be performed for that palette.

For example, the Adapter palette has activities that can publish messages to a specified adapter or invoke an operation by way of an adapter. There is also an FTP palette that can invoke the PUT and GET commands on an FTP server.

Activities are available to communicate with a variety of systems. There is also a general-purpose Java code activity that allows you to write and execute standard Java code to perform custom processing in your process definition.

A process definition can begin with a Start activity, but some palettes contain activities that can start a process. These activities, also known as *process starters*, begin a process when the specified event occurs. For example, the Mail palette has a Receive Mail activity. If this is placed in a process definition, it replaces the Start activity and a process instance is started when a mail message is received.

[Chapter 4, Activities, on page 47](#) describes activities and their use in process diagrams.

Transitions

Transitions describe the flow of processing in a process definition. A transition is represented by an arrow between two activities. The arrows are unidirectional, and you cannot draw a transition to a previously executed activity. Control flow in a process definition must proceed sequentially beginning with the Start activity (or a process starter) and ending with the End activity.

A transition can optionally specify a condition. The condition determines if the transition is taken when an activity completes processing. After an activity completes, all transitions whose conditions are met are taken. You can have transitions from one activity to many other activities. Therefore, you can have several branches of activity processing in your diagram.



Having multiple branches in a process definition does not imply that each branch is processed concurrently. Transitions describe control flow of the process definition, not the concurrency of execution of activities. Process execution is controlled by the process engine. See *TIBCO ActiveMatrix BusinessWorks Administration* for more information about configuring the TIBCO ActiveMatrix BusinessWorks process engine.

Each activity in a process definition must have a transition to it, or the activity is not executed when the process executes.

[Chapter 5, Transitions and Conditions, on page 71](#) describes transitions and conditions.

Groups

Groups are used to specify related sets of activities. The main uses of groups are the following:

- To create a set of activities that have a common error transition. Basically, this is similar to a `try . . . catch` block in Java. This allows you to have a set of activities with only one error-handling transition, instead of trying to individually catch errors on each activity.
- To create sets of activities that are to be repeated. You can repeat the activities once for each item in a list, until a condition is true, or if an error occurs.
- To create sets of activities that participate in a transaction. Activities in the group that can take part in a transaction are processed together, or rolled back, depending upon whether the transaction commits or rolls back.

[Chapter 6, Grouping Activities, on page 77](#) describes groups and how to use them in a process definition.

Shared Configuration Resources

Shared configuration resources are specifications that are shared among activities. These are resources, such as database connections, WSDL files, schema definitions, and connections to other servers. Shared configuration resources are created outside of process definitions, but they are used when specifying the Configuration tab of some activities.

TIBCO ActiveMatrix BusinessWorks Palette Reference describes shared configuration resources.

Subprocesses

Business processes are often very complex and it is difficult to diagram the complete process in one process definition. You can create several smaller process definitions instead of one monolithic process definition. You can then call each process definition from another process definition, when necessary. When you call a process definition, the called process is known as a *subprocess*. Using subprocesses helps to make more readable process diagrams and you can reuse subprocesses across many process definitions.



Subprocesses cannot have process starters. That is, they must begin with the Start activity, not any activity that receives an event from an outside source.

To create and call a subprocess, follow this procedure:

1. Create a process definition for the subprocess. See [Developing Process Definitions on page 45](#) for more information about creating process definitions.
2. Specify the input, output, and any error schemas of the subprocess on the Start and End activities in the subprocess. See [Start Activity on page 67](#) and [End Activity on page 69](#) for more information about specifying the input, output, and error schemas of a process.
3. Create a process definition that will call the subprocess.
4. Place a Call Process activity (located on the General Activities palette) in the process definition.

The Call Process activity allows you to map input values into the called process, and optionally allows you to spawn the subprocess into another process instance.

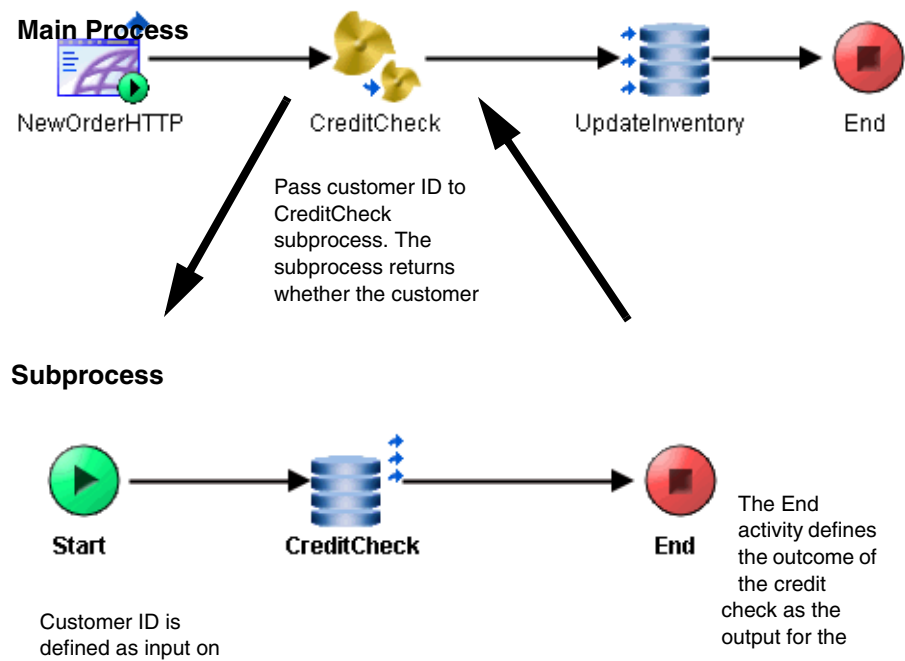


You can call specific processes, or you can dynamically determine which process to call when the process instance executes. See *TIBCO ActiveMatrix BusinessWorks Palette Reference* for more information about dynamically calling subprocesses.

Normally, a subprocess executes in the same process instance as the calling process, and the output of the subprocess is available to all subsequent activities in the process. If you select the checkbox in the Spawn field of the configuration tab of the Call Process activity, the subprocess is spawned into a new process instance. When a subprocess spawns a new process instance, the parent process cannot access the called process' output.

Figure 16 illustrates a main process calling a subprocess.

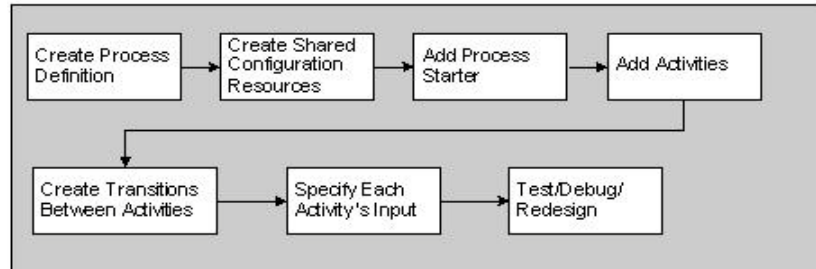
Figure 16 A main process calling a subprocess



Developing Process Definitions

Figure 17 describes the general procedure for developing process definitions.

Figure 17 *Developing process definitions*



The following is a more detailed description of how to develop process definitions:

1. Create one or more process definitions by dragging Process Definition resources from the process palette to the design panel. Name each process definition and give the process a description.
2. Create any Shared Configuration resources you will require for your process definition. These items are located in the Shared Configuration palette.

For example, if you are going to connect to a database, you should drag and drop a JDBC Connection into the design panel. This describes the username, password, JDBC URL and other information about the connection. You can then use this connection in many process tasks that require a connection to a database.

See *TIBCO ActiveMatrix BusinessWorks Palette Reference* for more details about each of the shared resources.

3. Select one of the process definitions you created in Step 1 in the project panel. This causes a blank business process to appear in the design panel. The business process has a Start and an End activity by default.

See [Chapter 4, Activities, on page 47](#) for more information about activities.

4. Select the activities that start the business process. These activities are known as event sources or process starters. Drag these event sources from their palettes into the design panel.

For example, if you wish to start a process when an HTTP request is received, select the HTTP activity palette, locate the HTTP Receiver process starter activity, and drag and drop it into the design panel.

5. Configure the process starter activities. See the documentation for the activity you are configuring for more information about the fields on each of the tabs for the activity.
6. Drag and drop more activities to define the business process.
7. Click the transition button on the tool bar to draw transitions between activities.

See [Chapter 5, Transitions and Conditions, on page 71](#) for more information about specifying transitions.

8. Perform mapping and transformation of data for each activity's input.
See [Chapter 8, Mapping and Transforming Data, on page 117](#) for more information about mapping and transforming data.
9. Once the process definition is complete, use the test mode tool to enter testing mode.

In testing mode, a TIBCO ActiveMatrix BusinessWorks engine is started to perform the processing described in the process definition. Test and debug your process definition until it operates as expected.

See [Chapter 18, Testing Process Definitions, on page 311](#) for more information about testing process definitions.

Chapter 4 **Activities**

Activities perform the work in a process definition. This chapter describes activities and how to use them in a process definition.

Topics

- [Activity Overview, page 48](#)
- [Activity Icons, page 50](#)
- [Configuration, page 52](#)
- [Advanced, page 53](#)
- [Event, page 54](#)
- [Editor, page 56](#)
- [Input, page 62](#)
- [Output, page 64](#)
- [Process Starters, page 65](#)
- [Start Activity, page 67](#)
- [End Activity, page 69](#)

Activity Overview

Activities are the individual units of work in a process definition. Activities are generally operations that interface to external systems, but activities can also perform internal processing. Activities are available on the various palettes in TIBCO Designer. Each palette has a set of activities that can be performed for that palette. For example, the following activities are included in the ActiveEnterprise Adapter palette:

Table 4 Example activities

Activity Name	Function
Publish to Adapter	Sends a message to the specified adapter.
Adapter Subscriber	Receives a message from the specified adapter and starts a process.
Respond to Adapter Request	Replies to a message sent by an adapter.

The following are examples of palettes and some of the activities the palettes contain:

- File
 - Create File
 - Remove File
 - Write File
 - Read File
- FTP
 - FTP Put
 - FTP Get
- JDBC
 - JDBC Query
 - JDBC Call Procedure
 - JDBC Update
- Mail
 - Send Mail

When you are in a process definition in TIBCO Designer, the activity palettes are available to drag and drop activities into the process definition. Each activity usually has two or more of the following tabs for specifying the characteristics of the activity:

- **Configuration** — Used for general configuration of the activity. For example, this tab specifies the adapter service to use for an Adapter activity.
- **Advanced** — Any advanced configuration parameters are specified here.
- **Event** — For activities that wait for incoming events, such as HTTP requests or incoming TIBCO Rendezvous messages, this tab specifies the timeout for the incoming event and a condition to determine whether the incoming event is the correct one for the specific process instance.
- **Editor** — A data schema for the activity. This is used when the input or output data is not known by the activity, and the user must specify their own schema. Once specified, the schema becomes available on the Input, Output or both tabs of the activity.
- **Input** — The output data from all activities that precede this activity in the process definition is available for mapping to this activity's input schema.
- **Output** — The activity's data is output to activities that follow in the process definition.

The sections that follow describe each tab used to specify an activity. There is a chapter for each available activity palette. See the activity palette chapter for more information about the specific activity you wish to use.

There are two activities that are included in a process definition by default: the Start activity and the End activity. See [Start Activity on page 67](#) and [End Activity on page 69](#) for more information about these activities.

Activity Icons

Each activity is represented by an icon in the palette and design panels. These icons represent the function of the activity. There are some common elements of activity icons that represent the type of activity and the direction of data between the activity and the process.

Table 5 describes the various elements in activity icons.

Table 5 Activity icon elements

Element	Example	Description
	 Publish to Adapter	Arrows indicate the direction of information between the process and the external system. Multiple arrows indicate either sending or receiving data, as opposed to performing an operation (see the description of single arrows below). In the example, the information is flowing from the process to the adapter (the adapter is represented by the purple and blue sphere).
	 Adapter Request-Response Server	A green circle with an arrow inside (similar to a "Play" button on a media player) indicates the activity is a process starter. These activities start new processes based on the receipt of an event from the external system. See Process Starters on page 65 for more information about process starters.
	 Wait for Adapter Message	A yellow square with two parallel lines inside (similar to a "Pause" button on a media player) indicates the activity waits for an incoming event from the external system. These activities, also known as "Event activities" cause the process to suspend until the incoming event is received. See Event on page 54 for more information about Event activities.

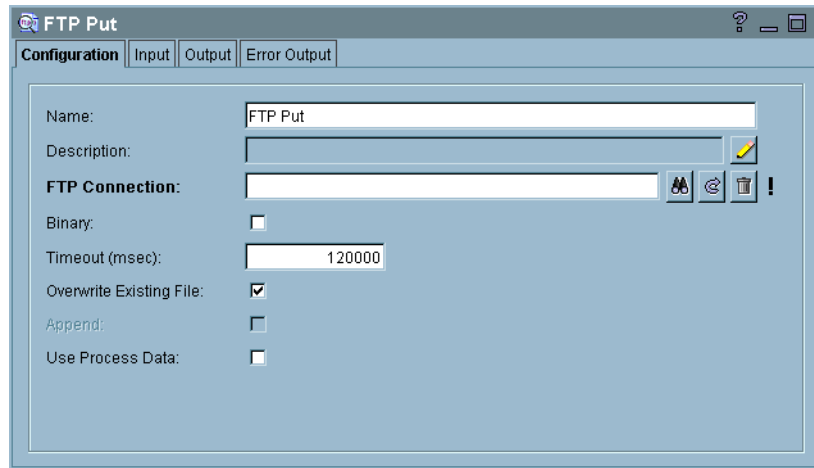
Table 5 Activity icon elements

Element	Example	Description
	 Invoke an Adapter Request-Response Service	A single arrow going into or out of the external system indicates that the activity is performing a request, sending a response, or performing a request and receiving a response. This is different from simply receiving a message or data (indicated by multiple arrows) because the activity is performing or responding to a remote operation call. The direction of the arrow indicates whether the activity is receiving a request, sending a response, or sending a request and receiving a response. In the Invoke an Adapter Request-Response activity example, the activity is sending a request to an adapter and expects to get a response from the adapter.
	 Adapter Request-Response Server	In the Adapter Request-Response Server activity example, the activity starts a process based on the receipt of a request from an adapter.
	 Respond to Adapter Request	In the Respond to Adapter Request activity example, the activity is sending a response to a previously received adapter request.

Configuration

The configuration tab contains the general specifications for the activity. For example, an FTP activity would contain specifications for the FTP session, such as whether the type of data being sent is text or binary or whether the FTP server resides outside of a firewall. Required fields on the configuration tab are displayed in bold so that it is easy to see the minimum required information for configuration of an activity.

The following illustrates the configuration tab.



In general, all activities allow you to specify a name for the activity and provide a short description on the Configuration tab. Any other items on the configuration tab are the required configuration elements you must specify to make the activity work.

See the chapter for the palette you are interested in *TIBCO ActiveMatrix BusinessWorks Palette Reference* for more information about the Configuration tab of the activities in that palette.

Advanced

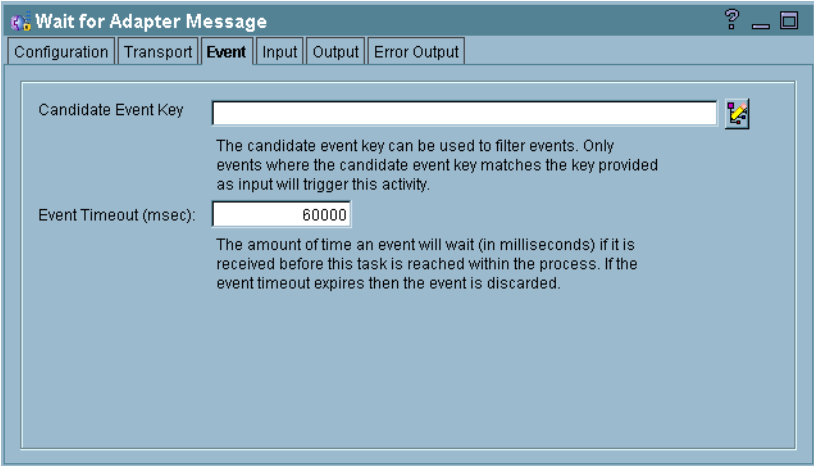
The Advanced tab is available on some activities for specifying additional configuration options. See the chapter for the palette you are interested in *TIBCO ActiveMatrix BusinessWorks Palette Reference* for more information about the Advanced tab of the activities in that palette.

Event

The Event tab is available on activities that expect an incoming event. These are activities that wait for an incoming event in a process. These activities cause the process instance to suspend until the incoming event is received. An Event tab has the following fields:

Field	Description
Candidate Event Key	Expression used to evaluate whether the incoming message is appropriate for this process. This expression is specified in XPath, and only data from the incoming event is available for use in this XPath expression. See Chapter 9, XPath, on page 161 for more information about XPath expressions.
Event Timeout (msec)	The amount of time a message will wait (in milliseconds) if it is received before this task is reached in the process. If the event timeout expires, an error is logged and the event is discarded. If no value is specified in this field, the message waits indefinitely. If zero is specified, the event is discarded immediately, unless this has already been reached.

The following illustrates the Event tab.



[Table 6](#) describes the available activities with Event tabs. See the chapter for the palette you are interested in for more information about tasks that have Event tabs.

Table 6 Activities with Event tabs

Palette	Event Activities	Waits For...
ActiveEnterprise Adapter	Wait for Adapter Message Wait for Adapter Request	An adapter message or request.
File	Wait for File Change	The specified file to change.
General Activities	Wait	The associated Notify activity to execute.
HTTP	Wait for HTTP Request	An HTTP request.
JMS	Wait for JMS Queue Message Wait for JMS Topic Message	Either a JMS queue or topic message.
Manual Work	Wait For Completion	A manual task to complete.
Rendezvous	Wait for Rendezvous Message	A TIBCO Rendezvous message.

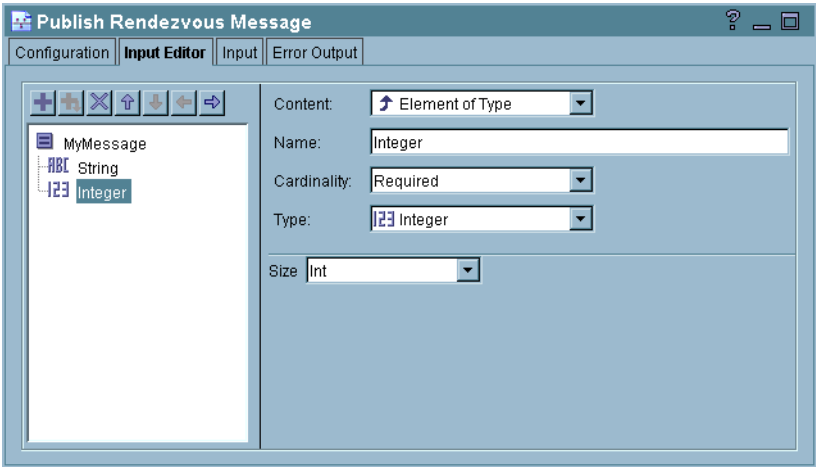
Editor

The Editor tab is used to specify a data schema for input or output of an activity. This tab is useful when the data does not have a well-known structure. The data schema may also be for a more specialized use, such as for defining the output headers of an incoming HTTP request. The name of the Editor tab differs depending upon what the schema is used for.

For example, the Input Editor tab of the Publish Rendezvous Message activity allows you to define the schema for the message you wish to publish.

You can use a simple datatype, or you can define a group of data elements on this tab. You can also reference XML schema or ActiveEnterprise classes stored in the project. Once defined, the schema appears on the appropriate tabs of the activity. The data in the schema then becomes available to other activities in the process definition.

The following illustrates the Schema tab. In this example, the Schema tab is labeled Input Editor indicating this defines the schema for the activity’s input.



To define a schema on this tab, use the buttons above the schema tree to add, delete, or move data items. Then use the fields of the dialog to specify the datatype of each item.

Field	Description
Content	Defines the content of the element. The other fields that appear depend upon which content type is selected. See Table 7 for more information about the content type.

Field	Description
Name	The name of the element.
Cardinality	The qualification for the data item. Data items can be specified as one of the following: • Required — the data item is required and must be supplied when the process is called. • Optional (?) — the data item is optional. • Repeating, Zero or More (*) — The data item is a list that has zero or more elements. • Repeating, One or More (+) — The data item is a list that has one or more items.
Type	The type of data. Can be any of the following: • XML Type Reference — must locate the stored XML schema definition. • Any of the datatypes described in Table 8.
Schema Name	Stored XML schema that contains the element or type you wish to reference.
Type Name	Type in a stored XML schema you wish to reference.

[Table 7](#) describes the potential content types for data elements.

Table 7 Content types for schema elements

Content Type	Description	Other Fields for This Content Type
Complex Element	An element that contains other elements. This is like a structure in a programming language. The complex element can contain zero or more elements of other types, and it can contain other complex elements.	Name Cardinality

Table 7 Content types for schema elements

Content Type	Description	Other Fields for This Content Type
Element of Type	An element with a specified datatype. You can specify a scalar datatype (string, integer, and so on), you can reference an XML type, or specify the TIBCO ActiveEnterprise Any datatype.	Name Cardinality Type Other fields depending upon the datatype selected
XML Element Reference	A reference to an element in a stored XML schema. See TIBCO Designer documentation for more information about XML schema.	Cardinality Schema Element
Attribute of Type	An attribute with a specified datatype. You can specify a scalar datatype (string, integer, and so on), you can reference an XML type, or specify the TIBCO ActiveEnterprise Any datatype.	Name Cardinality Type Other fields depending upon the datatype selected.
Sequence	A sequence of elements. Each item in the sequence is a structure of the sub-elements of this element.	Cardinality
Choice	A choice of elements. The datatype of this element can be one of the sub-elements defined.	Cardinality
All	The datatype of this element can be all of the datatypes of the sub-elements defined.	Cardinality
XML Group Reference	A reference to an XML group in a stored XML schema. See TIBCO Designer documentation for more information about XML schema.	Cardinality Schema Model Group

Table 7 Content types for schema elements

Content Type	Description	Other Fields for This Content Type
Any Element	A reference to any XML Element. You can use the Coercions button to supply a reference to the XML Element for this item when it appears in the input or process data.	Cardinality Validation
WSDL Message	A reference to a message defined in a WSDL File resource. Use the Browse button to bring up a dialog for locating WSDL files within the project.	WSDL Message

Table 8 describes the datatypes available for data.

Table 8 Icons for schema datatypes

Icon	Description
	String or character value. You can specify the type of string as one of the following: - String - Normalized String - Token - Language - Name - NC-Name - Q-Name - Name Token - Name Tokens - ID - ID ref - ID refs - Entity - Entites

Table 8 Icons for schema datatypes

Icon	Description
	Integer value. You can specify the size of the integer as one of the following: • Byte • Short • Int • Long • Unsigned Byte • Unsigned Int • Unsigned Long • Integer • Positive Integer • Negative Integer • Non-positive Integer • Non-negative Integer
	Floating point number. You can specify the size of the schema item as float, double, or decimal.
	Boolean value.
	Date or Time. This can be any of the following datatypes: • Time • Date • Date & Time • Duration • Day • Month • Year • Year & Month • Month & Day

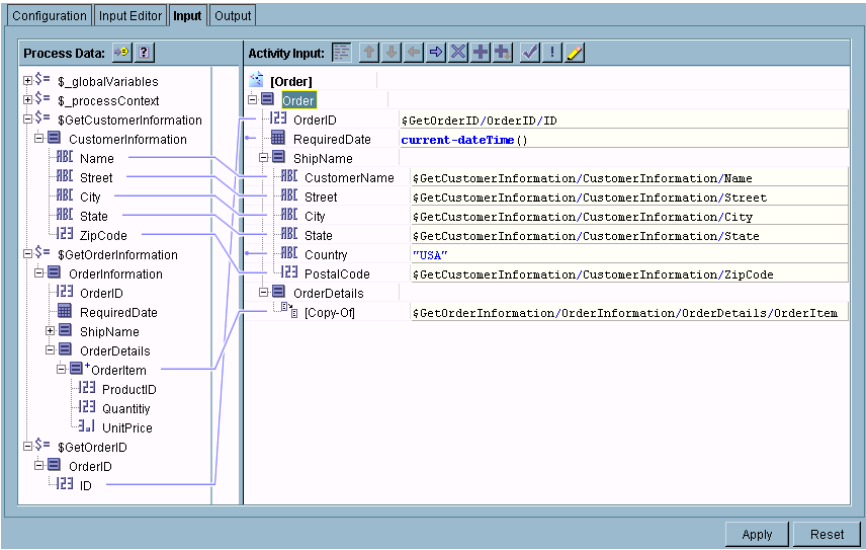
Table 8 Icons for schema datatypes

Icon	Description
	Base 64 or hexadecimal value.
	An HTTP Uniform Resource Identifier.
	Complex element. Container for other datatypes.
	XML element or group reference.
	Sequence. Signifies that the contained sub-elements are repeated in an ordered sequence.
	Any Type. Represents a schema element with the TIBCO ActiveEnterprise datatype any. This element can be specified as any other datatype or a reference to an XML Type or AE Class. You can use the Coercions button to supply a datatype for this element.
	Any Element. Represents a schema element that can be a reference to any XML Element. You can use the Coercions button to supply a reference to the XML Element for this element.
	Choice. Specifies that the schema element can be one of a specified set of datatypes.

Input

The Input tab allows you to map and transform output data from the previous activities in the process (including the event that starts the process) to input data for the activity. The Process Data area contains the output from all of the activities that appear prior to the activity in the process definition. The Activity Input area lists the current activity's required and optional input data.

The following illustrates the input tab.



Mapping and Transforming Activity Input Data

You can create mappings between the available output from previous activities and the current activity's input. You can also specify constants (strings enclosed in quotes or numbers) for any input values, or you can specify conditions on the mappings.

In general, to create a mapping, you click on the desired item in the available schema in the Process Data panel and drag the item to the desired item in the Activity Input panel. If you wish to type in a constant or expression, you can click on the schema item in the Activity Input panel and type the constant or expression into the field.

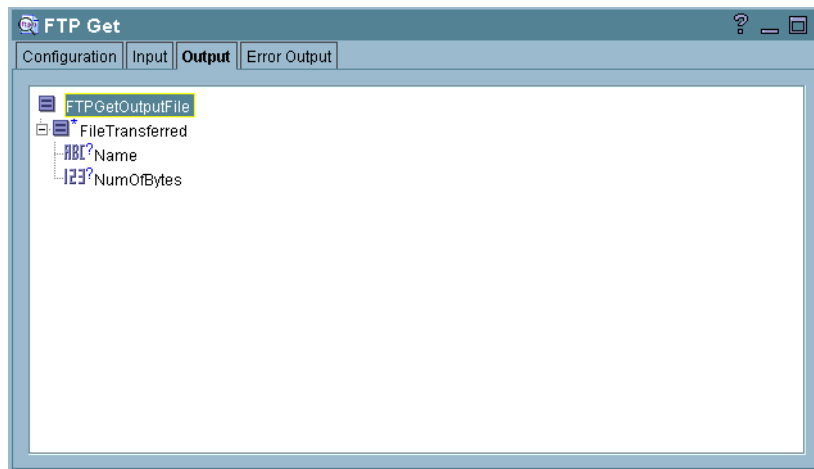
There are also several icons above the Activity Input area. [Table 11 on page 121](#) describes the icons and their function.

See [Chapter 8, Mapping and Transforming Data, on page 117](#) for more information about using the Input tab to create mappings between process data and the activity's input.

Output

The output tab displays the activity output schema. This name appears in subsequent activities input tabs. The activity output data is displayed for informational purposes only and cannot be modified or altered.

The following illustrates the output tab.



Process Starters

Some activities are used to start a process when an event occurs. For example, in the File palette, there is an activity named File Poller. This activity detects changes in a specified file and starts a process when the change occurs. This kind of activity is known as a *process starter*. When a process starter is placed into a process definition, it replaces the default Start activity, and becomes the first activity in the process.

You can only have one process starter in a process definition. You will receive a warning if you attempt to add more than one process starter to a process definition. [Table 9](#) describes the available process starters.

When you deploy your project, you can place processes with different process starters on different machines. For example, you may wish to place all processes with a Receive Mail process starter on the same machine as the mail server so that the processes can poll the mail server more efficiently. See *TIBCO ActiveMatrix BusinessWorks Administration* for more information about deployment and specifying where process starters run.

Table 9 Process starters

Palette	Process Starter	Starts a process when...
ActiveEnterprise Adapter	Adapter Subscriber Adapter Request-Response Server	A message or a request from an adapter is received.
File	File Poller	The specified file is created, changed, or deleted.
General Activities	Timer	The specified time interval occurs. You can start processes for one-time events or schedule processes to start on recurring time intervals.
	Receive Notification	A corresponding Notify activity has executed.
HTTP	HTTP Receiver	An HTTP request is received.
JMS	JMS Queue Receiver	Either a JMS queue or topic message is received.
	JMS Topic Subscriber	

Table 9 Process starters

Palette	Process Starter	Starts a process when...
Mail	Receive Mail	Mail for the specified user is received by the mail server.
Rendezvous	Rendezvous Subscriber	A TIBCO Rendezvous message is received.
SOAP	SOAP Event Source	A SOAP web services request is received.

Misc Tab

All process starters have a Misc tab that allows you to configure features common to all process starters. The Misc tab contains the following fields:

Field	Description
Sequencing Key	This field can contain an XPath expression that specifies which processes should run in order. Process instances whose sequencing key evaluates to the same value will be executed sequentially in the order in which the process instance was created. See Chapter 12, Process Instance Execution, on page 207 for more information about controlling the execution order of process instances. See Chapter 9, XPath, on page 161 for more information about XPath expressions.
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 <code>\$_processContext</code> process variable. See Chapter 9, XPath, on page 161 for more information about XPath expressions. See Chapter 7, Working With Variables, on page 105 for more information about process variables.

Start Activity



Start

The Start activity is the first activity in a process definition (process starters replace the Start activity when they are used in a process definition).

A process can be called from another process, and the Start activity is used to define the input expected by the process. The Start activity has the following tabs:

- [Configuration](#)
- [Output Editor](#)
- [Output](#)

See [Subprocesses on page 43](#) for more information about calling a process from another process.

Configuration

The configuration tab has the following fields.

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

Output Editor

The Output Editor tab defines the data that the process is expecting as input. Any process that calls this process definition must supply the data specified on the Output Editor tab.

You can define your own datatype on this tab, and you can reference XML schema or ActiveEnterprise classes stored in the project. Once defined, the data specified on the Output Editor tab becomes the output schema of the Start activity. This data then becomes available to other activities in the process definition.

See [Editor on page 56](#) for a description of how to define a schema.

Output

The output for the activity is defined by the specified data elements on the Output Editor tab.

End Activity



The End activity is the last activity in a process definition. When a process definition is called from another process, you may wish to have the called process definition output data to the calling process. You can map data from the activities in the process to an output schema specified on the End process. This becomes the output of the process.

The End activity has the following tabs:

- [Configuration](#)
- [Input Editor](#)
- [Input](#)
- [Error Schemas](#)

See [Subprocesses on page 43](#) for more information about calling a process from another process.

Configuration

The configuration tab has the following fields.

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

Input Editor

The Input Editor tab defines the data that the process will output. Any process that calls this process definition will receive this data when the process call completes.

You can define your own datatype on this tab, and you can reference XML schema or ActiveEnterprise classes stored in the project. Once defined, the data specified on the Input Editor tab becomes the input schema of the End activity. You can then map data from other activities in the process to the End activity's input, and this becomes the output of the process when the process completes.

See [Editor on page 56](#) for a description of how to define a schema.

Input

The input for the activity is defined by the specified data elements on the Input Editor tab.

Error Schemas

The Error Schemas tab defines schemas to contain data for errors thrown by the process definition. You can define multiple schemas, each for use in specific error cases.

You can define your own datatype on this tab, and you can reference XML schema or ActiveEnterprise classes stored in the project. See [Editor on page 56](#) for a description of how to define a schema.

See [Chapter 10, Error Handling, on page 174](#) for more information on error handling.

Chapter 5

Transitions and Conditions

Transitions and conditions control the flow of activities in a process diagram. This chapter explains how to create transitions and specify conditions on those transitions.

Topics

- [Transitions, page 72](#)
- [Conditions, page 75](#)

Transitions

Transitions describe the flow of processing. A transition is represented by an arrow between two activities or groups of activities in a process definition. The arrows are unidirectional, and you cannot draw a transition to a previously executed activity. Control flow in a process definition must proceed sequentially beginning with the starting activity and ending with the End activity. If you wish to perform looping, use groups to specify multiple executions of grouped activities (see [Chapter 6, Grouping Activities, on page 77](#) for more information on groups).

After an activity or group completes processing, all transitions whose conditions are met are taken. You can have transitions from one activity to many other activities. Therefore, you can have several branches of activity processing in your diagram.

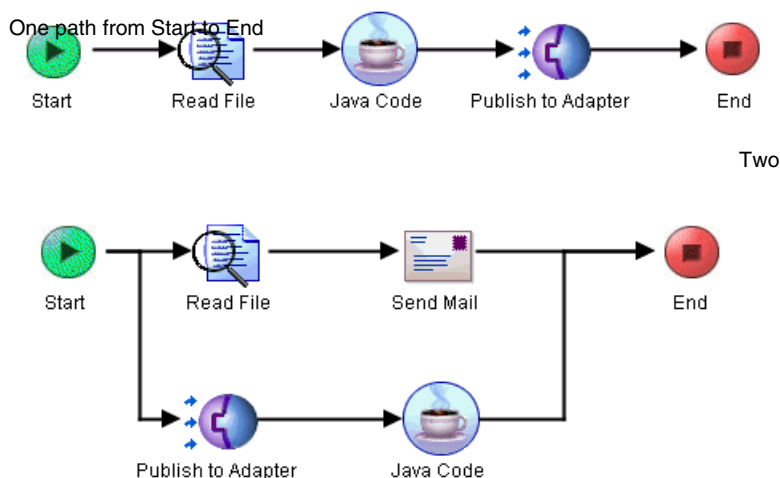


Having multiple branches in a process definition does not imply that each branch is processed concurrently. Transitions describe control flow of the process definition, not the concurrency of execution of activities. Process execution is controlled by the process engine. See *TIBCO ActiveMatrix BusinessWorks Administration* for more information about configuring the TIBCO ActiveMatrix BusinessWorks process engine.

Each activity in a process definition must have a transition to it, or the activity is not executed when the process executes.

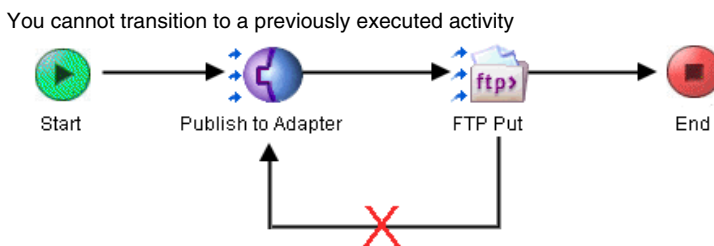
[Figure 18](#) illustrates examples of valid transitions in a process. [Figure 19](#) illustrates an invalid transition.

Figure 18 Valid transitions



If you have multiple branches in a complex process definition, you may find the Null activity in the General Activities palette useful for joining the multiple branches into a single execution path. See *TIBCO ActiveMatrix BusinessWorks Palette Reference* for more information about the Null activity.

Figure 19 An invalid transition



A transition is taken depending upon the condition specified on the transition. When a transition is created, you may specify whether the transition is taken always, only when an error is encountered, only if no other transitions are taken, or when a custom-specified condition based on process data values is true. See [Conditions on page 75](#) for more information.

Creating a Transition

To create a transition, follow this procedure:

1. First create or open a process definition that contains at least two activities.
2. Click on the Create Transition icon  on the TIBCO Designer toolbar.
3. Position the cursor over the first activity.
4. Click and hold the mouse button.
5. Drag the mouse until the cursor is positioned over the activity that you would like to transition to.
6. Release the mouse button.

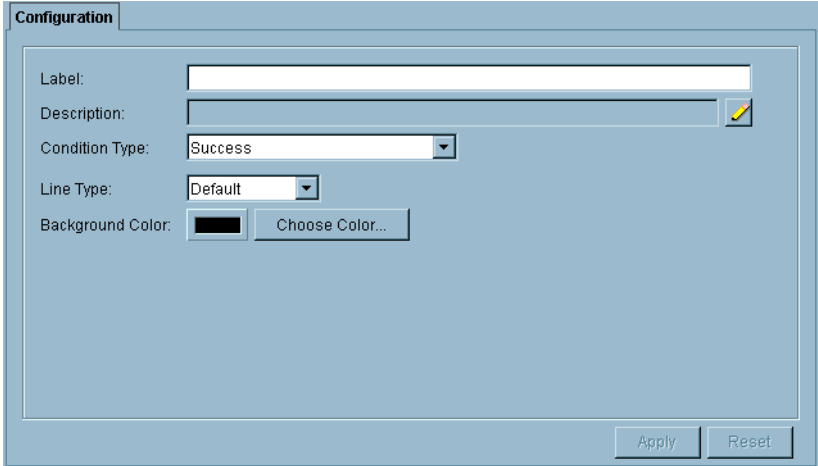
Once the transition is created, the condition dialog is presented. The condition dialog allows you to specify when this transition is taken. See [Conditions on page 75](#) for more information about specifying conditions.

7. Optionally, choose the line type and color for the transition. You can choose any of the following line types for your transition:
 - Default (straight line)
 - Straight
 - Multiple Bends
 - One Bend
 - Curved
 - S-Shaped

Conditions

Conditions are specified on transitions to determine whether to take the transition to the next activity or not. When a transition is created, you are presented with the transition dialog. [Figure 20](#) illustrates the transition dialog.

Figure 20 The transition dialog



The transition dialog allows you to specify a label, description, line type, and background color for the transition. You can also specify a condition type for the transition. The following table describes each condition type.

Condition Type	Description
Success	Take this transition unconditionally. That is, always transition to the activity the transition points to, if the activity completes successfully. This is the default condition for transitions.

Condition Type	Description
Success with condition	Specify a custom condition using XPath. If the activity completes successfully, and the condition you create evaluates to true, the transition is taken to the activity it points to. You can type in an XPath condition, and you can use the XPath formula builder to drag and drop XPath expressions and data into the condition. See Chapter 9, XPath, on page 161 for more information about specifying XPath conditions and using the XPath formula builder.
Success if no matching condition	Take this transition when the activity completes successfully, but only if no other transitions are taken. This is useful when multiple transitions with conditions are drawn to other activities. This condition type can be used to handle any cases not handled by the conditions on the other transitions.
Error	Take this transition if there is an error during processing of the activity. See Chapter 10, Error Handling, on page 174 for more information on error handling in process definitions.

When a transition is created, the default condition type is Success. If a condition other than Success is specified, it is displayed on the transition line in the process definition if a label is not specified in the Label field.



There can be only one Error and one Success if no matching condition transition out of each activity.

Chapter 6

Grouping Activities

This chapter describes groups and how to use them for transactions, error-handling, and looping.

Topics

- [Overview of Groups, page 78](#)
- [Group Configuration Tab, page 82](#)
- [No Action Groups, page 88](#)
- [If Groups, page 89](#)
- [Critical Section Groups, page 90](#)
- [Pick First Groups, page 92](#)
- [Overview of Loops, page 93](#)
- [Iterate Loop, page 95](#)
- [Repeat Until True Loop, page 98](#)
- [While True Loop, page 100](#)
- [Repeat On Error Until True Loop, page 101](#)

Overview of Groups

Groups are used to specify related sets of activities. The main uses of groups are the following:

- To create a set of activities that have a common error transition — similar to a `try...catch` block in Java. This allows you to have a set of activities with only one error-handling transition, instead of trying to individually catch errors on each activity. See [No Action Groups on page 88](#) for more information.
- To create sets of activities that participate in a transaction. Activities in the group that can take part in a transaction are processed together, or rolled back, depending upon whether the transaction commits or rolls back. See [Chapter 11, Transactions, on page 185](#) for more information about transactions.
- To create sets of activities that are executed conditionally, such as in an `if ... then ... else if ...` construct in a programming language.
- To create sets of activities that are to be repeated. You can repeat the activities once for each item in a list, while or until a condition is true, or if an error occurs. See [Overview of Loops on page 93](#) for more information about loops.
- To create a critical section that synchronizes process definitions. See [Critical Section Groups on page 90](#) for more information about critical sections.
- To specify that the first activity that completes should determine which transition(s) to take to continue processing. This allows you to wait for one or more incoming events and continue processing based on what incoming event was received first. See [Pick First Groups on page 92](#) for more information about this type of group.

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 a process definition 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, perform the following procedure:

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.

4. The group configuration appears in the configuration panel.
5. Specify the type of group to create and any other configuration parameters required for the group. See [Table 10](#) for more information about the fields of the group configuration tab.
6. 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.
7. 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, perform the following procedure:

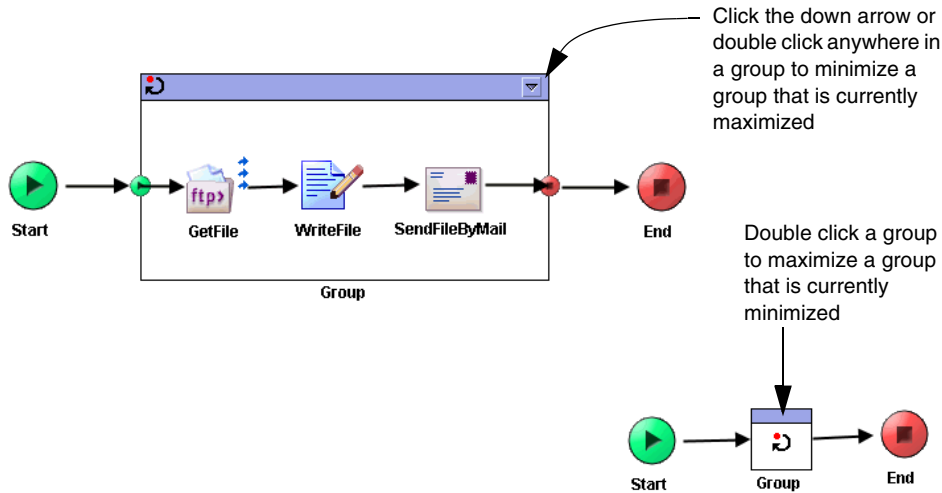
1. Choose the Select tool (the arrow pointer in the tool bar).
2. Select the group in the design panel.
3. Choose **View > Remove a Group** from the menu, or click the Undo the group button. 

To minimize or maximize the display of a group, perform the following procedure:

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 21](#) illustrates minimizing and maximizing a group.

Figure 21 Minimizing and maximizing groups



To resize a maximized group, perform the following procedure:

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 executed activities inside or outside the group. If the group is used for a loop (iterate, repeat until true, and so on), activity output in the group 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 process definition. In the case of loop groups, only output from the last execution of the activity is available.

If you wish to store data from each successive iteration of a loop, you can create a process variable and use the Assign activity in the loop to store or alter data for each iteration. For Iterate and Repeat Until True loops, you can optionally accumulate the output of each execution of one activity in the group into a list. This list becomes the group's output and the list is available to subsequent activities in the process definition.

Group Configuration Tab

Table 10 describes the fields in the Configuration tab for groups.

Table 10 Group Configuration tab

Field	Description
Name	The name to appear as the label for the group in the process definition.
Description	Short description of the group.
Group Action	The type of group. Groups can be of the following types. • None — See No Action Groups on page 88 for more information. • If Groups — See If Groups on page 89 for more information. • Iterate Loop — See Iterate Loop on page 95 for more information. • Repeat Until True Loop — See Repeat Until True Loop on page 98 for more information • While True Groups — See While True Loop on page 100 for more information. • Repeat On Error Until True Loop — See Repeat On Error Until True Loop on page 101 for more information. • Transaction Groups — See Chapter 11, Transactions, on page 185 for more information. • Critical Section — See Critical Section Groups on page 90 for more information. • Pick First — See Pick First Groups on page 92 for more information.

Table 10 Group Configuration tab

Field	Description
2.x Compatibility Mode	This field is only available for loop groups in process definitions that were migrated from TIBCO ActiveMatrix BusinessWorks 2.x. Checking this field indicates that you would like activity output in a group to be handled using the TIBCO ActiveMatrix BusinessWorks 2.x semantics. In 2.x, activity output was not reset at the start of every iteration of a loop. Therefore, activities in a loop could potentially see activity output from previous iterations of the loop. If you uncheck this field, activity output in loop groups behaves as described in Activity Output and Groups on page 80.
Group Action: Iterate	
Index Name	The index variable for the loop. This variable will be 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 93 for more information.
Variable List	A process variable 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 choose from a list of available process variables for this field. Once a process variable is chosen, the correct XPath expression for that process variable is automatically entered into 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 96 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 93 for more information.

Table 10 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: Repeat-Until-True	
Index Name	The index variable for the loop. This variable will be 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 93 for more information.
Conditions	The condition that specifies when the loop should stop. The activities in the group are executed once, then the condition is checked. If the condition evaluates to false, the loop repeats, if the condition evaluates to true, the loop stops. The loop continues to repeat until the condition evaluates to true. The condition is specified as an XPath expression and the XPath formula builder is available to help to create the condition. See Chapter 9, XPath, on page 161 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 93 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 process variable to store the successive output of the selected activity in the Output Activity field.

Table 10 Group Configuration tab

Field	Description
Group Action: While True	
Index Name	The index variable for the loop. This variable will be 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 93 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 Chapter 9, XPath, on page 161 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 93 for more information.
Group Action: Repeat-On-Error-Until-True	
Index Name	The index variable for the loop. This variable will be 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 93 for more information.

Table 10 Group Configuration tab

Field	Description
Conditions	The condition that specifies when the loop should stop. The activities in the group are executed once. If an error occurs during the processing of the activities, and that error does not have an associated error transition, the condition is checked. If the condition evaluates to false, the loop repeats, if the condition evaluates to true, the loop stops. The loop continues to repeat if unhandled errors are encountered, until the specified condition evaluates to true. The condition is specified as an XPath expression and the XPath formula builder is available to help to create the condition. See Chapter 9, XPath, on page 161 for more information.
Suspend (If Still Error)	Suspends the process if the error still occurs when the specified condition is true. See Suspend If Still Error Option on page 102 for more information about this field.
Group Action: Transaction	
Transaction Type	Defines the type of transaction for group. • JDBC • Java Transaction API (JTA) User Transaction • XA Transaction • JMS Transaction
Include in Transaction	Specifies if the Receiver is included in the transaction, when JMS Transaction type is selected.

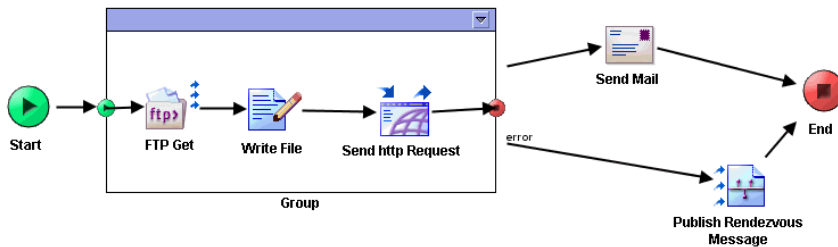
Table 10 Group Configuration tab

Field	Description
Group Action: Critical Section	
Scope	Defines the scope of the critical section group. • Single Group — specifies that all process instances in the same process engine for the current process definition will be synchronized on the current group. • Multiple Group — specifies that all process instances for the current process definition and any other process definition with a critical section group specifying the same lock resource will be synchronized. If you are synchronizing across multiple process engines, you should select the Multiple Group option. This option requires the Lock resource to be specified in the Lock Object field.
Lock Object	The Lock shared configuration resource that synchronizes critical section groups across process definitions and potentially across process engines.

No Action Groups

You can group a set of related activities, with a common set of transitions into and out of the group. If you do not wish for the activities in the group to repeat, specify the group action to be None. No action groups are primarily useful for specifying a single error transition out of the group so that if an unhandled error occurs in the group, you only need one error transition instead of an error transition for each activity. This behavior is similar to a `try . . . catch` block in Java.

The following process definition illustrates a no action group that has one error transition out of the group to process any unhandled errors that occur when the grouped activities execute.

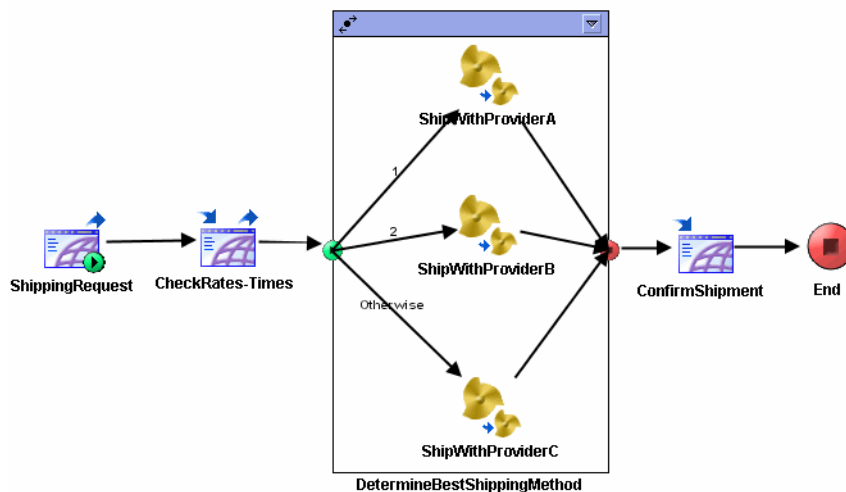


The process performs the following operations:

1. An FTP Get activity retrieves a file from an FTP server.
2. A Write File activity writes the retrieved file so that its contents are available at a later time.
3. The contents of the file are used to create an HTTP request to a web server.
4. In the event of an error, a TIBCO Rendezvous message is published so that an administrative application can be notified of the error. If all activities in the group succeed, an email message is sent.

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 a 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.

Critical Section Groups

Critical section groups are used to synchronize process instances so that only one process instance executes the grouped activities at any given time. Any concurrently running process instances that contain a corresponding critical section group wait until the process instance that is currently executing the critical section group completes.

Critical Section groups are particularly useful for controlling concurrent access to shared variables (see [Synchronizing Access to Shared Variables on page 116](#) for more information). However, other situations may occur where you wish to ensure that only one process instance is executing a set of activities at a time.

Synchronization Options

Critical section groups can be used to synchronize all process instances for a particular process definition in a single process engine, or you can synchronize process instances for multiple process definitions, or you can synchronize process instances across multiple process engines.

Single Group

If you wish to synchronize process instances for a single process definition in a single process engine, perform the following:

1. Create a group around the activities you wish to synchronize.
2. Specify Critical Section for the Group Action field.
3. Specify Single Group for the Scope field.

Only one process instance at any given time will execute the activities contained in the Critical Section group.

Multiple Groups

If you wish to synchronize process instances for multiple process definitions, or if you wish to synchronize process instances across multiple process engines, perform the following:

1. Create a Lock shared configuration resource and specify a name for the resource.
2. To perform the synchronization across multiple process engines, check the Multi-Engine field of the Lock resource.

When the process instances are executed by the same process engine, locking is performed in memory. When the process instances are executed across multiple engines, the process engines must be configured to use a database for storage, and a database transaction is performed to ensure that only one process instance is executing the critical section group at any given time.

3. Create a group around the activities you wish to synchronize.
4. Specify Critical Section for the Group Action Field.
5. Specify Multiple Groups for the Scope field.
6. Use the Browse button in the Lock Object field to locate the Lock shared configuration resource you created in Step 1.
7. Perform steps 3 to 6 for any process definitions you wish to synchronize. Make sure you specify the same Lock shared configuration object for all Critical Section groups.

Usage Guidelines

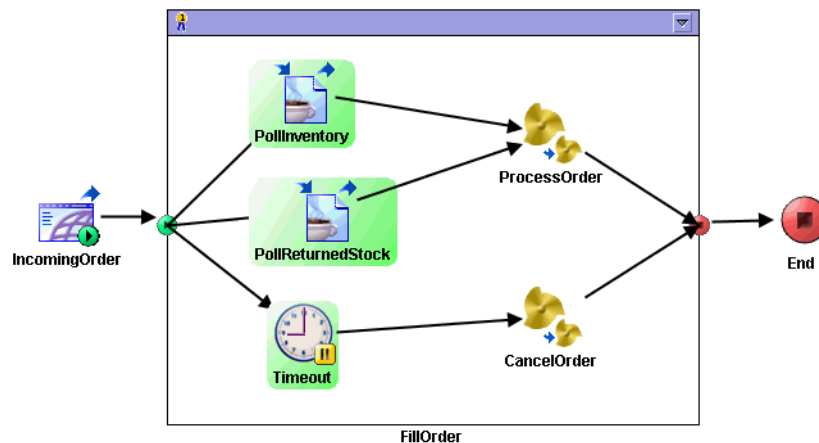
Because Critical Section groups cause many process instances to wait for one process instance to execute the activities in the group, there may be performance implications when using these groups. In general, you should use the following guidelines when creating critical section groups:

- Keep the duration of a Critical Section group as short as possible. That is, put only a very few activities in a Critical Section group, and only use activities that execute very quickly.
- Avoid nesting Critical Section groups. If you must use nesting, ensure that Lock shared configuration resources are used in the same order in all process definitions. Deadlocks can occur if you do not specify the Lock resources in the same order in nested Critical Section groups for all process definitions.
- Do not include any activities that wait for incoming events or have long durations, such as Request/Reply activities, Wait For activities, Sleep activities, or activities that require a long time to execute.
- When using Critical Section groups to retrieve or assign a value to a shared variable across multiple process engines, both the Lock and the Shared Variable shared configuration resources should have the Multi-Engine field checked. You can use a global variable to ensure that the Multi-Engine field is set to the same value for both resources.

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 example sub-process that uses the Pick First group to implement the business logic described above. The process is called and then a transition is taken to the Pick First group. The group then waits for either the return message from PollInventory, the return message from PollReturnedStock, or the Timeout activity. The first activity to complete determines the next transition taken. If either PollInventory or PollReturnedStock complete first, the transition to the ProcessOrder activity is taken. If the Timeout activity completes first, the transition to the CancelOrder 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 request/reply, Wait for... activities, and activities that have the pause symbol  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 the process data, you can iterate while or until a given condition is true, or you can iterate if an error is encountered while processing. The following are the types of loops that are available:

- [Iterate Loop](#)
- [Repeat Until True Loop](#)
- [While True Loop](#)
- [Repeat On Error Until True Loop](#)

Iterate and repeat until true 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 93](#) 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 data by referencing it with a dollar sign (\$) in front of it. For example, if the index variable is `i`, and you want to specify a condition that the loop should execute three times (for a repeat until true loop), the condition would be `$i=3`.



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

Accumulate Output

For iteration and repeat until true loops, 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 want to accumulate a list of customer names from repeated executions of a JDBC Database Query task, or you may wish to accumulate the sum of the amounts for line items in an order.

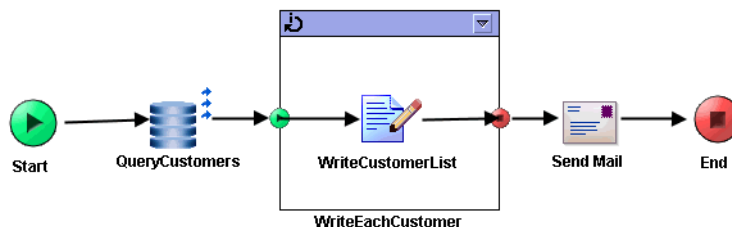


Because you can only select one activity in the Accumulate Output field, to accumulate output from several activities, you must use a different approach. One approach is to create a process variable to hold the data and use the Assign activity to assign values from each iteration of the loop to the process variable. Alternatively, you can use a Java Code activity to concatenate the data into the output parameters for the Java Code activity. You can then choose the Java Code activity as the Output Activity to accumulate for each iteration of the loop.

The output for the selected activity is accumulated each time the activity is executed. Therefore, if you choose to accumulate the output of the same activity used in the condition of a Repeat Until True loop, the activity is executed and the output is added to the list before the condition is checked. In this case, you may wish to use a Mapper activity in the loop to accumulate the output. The Mapper activity is placed after the activity used for the condition of the loop so that the loop exits before the value is accumulated. Alternatively, you can place a Mapper activity outside of the loop to strip out the unwanted value from the output list after the loop exits.

Iterate Loop

An Iterate loop repeats the series of grouped activities once for every item in an existing sequence or list. The list can be items of any datatype. The following is an example of an iterate loop.



The process performs the following operations:

1. A JDBC Query activity is used to query a database and populate a list of customer records. The customer records are then passed to a group containing one activity, WriteCustomerList.
2. The WriteCustomerList activity writes the name and address of each customer to a file, appending to the file as each record is written.
3. The group iterates once for every customer record returned by the QueryCustomer activity, and transitions to the ReadCustomerList activity once the last record is processed.
4. The process then reads the file that was written so that its data is available to the process, then transitions to a Send Mail activity to send the contents of the customer list by way of email.

The following is the configuration for this example:

The screenshot shows a 'Configuration' dialog box with the following fields and values:

- Name:** Group
- Description:** (empty)
- Group Action:** Iterate
- Index Name:** i
- Variable List:** \$QueryCustomer/Custom...
- Iteration Element:** current-record
- Accumulate output:** ☐

Buttons for 'Apply' and 'Reset' are located at the bottom right of the dialog.

In this example, the repeating element \$QueryCustomer/Custom... 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 records returned, the expression in the Variable List field would be the following:

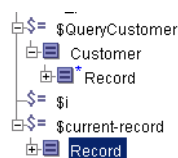
```
$QueryCustomer/Custom...[position() > 10]
```

See [Chapter 9, XPath, on page 161](#) 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,

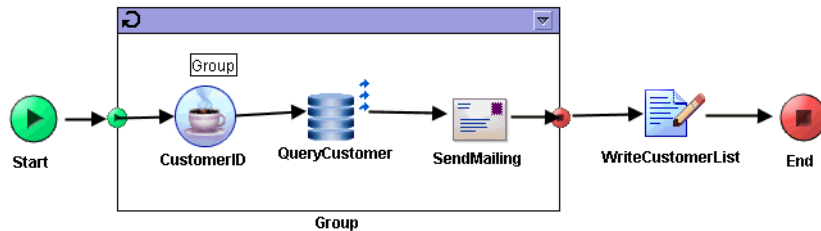
a process variable with the specified name appears in the Process Data tree in the Input tab. For example, in the Iterate group above, we specified `current-record` as the name for the current element. This causes the following to appear in the process data tree:



Notice that both `$QueryCustomer/Customer` and `$current-record` contain the element `Record`. `$current-record/Record` is a copy of the `$QueryCustomer/Customer/Record[i]` element, the current element being processed. If you use `$QueryCustomer/Customer/Record[i]` in an input mapping, TIBCO ActiveMatrix BusinessWorks traverses the `$QueryCustomer/Customer/Record` element to retrieve the current element each time the loop iterates. By using the `$current-record/Record` element instead, you save processes time in proportion to how many times the iteration loop repeats. The larger number of elements contained in the `$QueryCustomer/Customer/Record` repeating element, the greater the performance improvement you will notice by using `$current-record/Record` instead of `$QueryCustomer/Customer/Record[i]`.

Repeat Until True Loop

The Repeat Until True loop repeats the series of grouped activities until the given condition evaluates to true. The activities are always executed once before checking if the condition is true. After executing the series of activities, the condition is checked, and the loop exits when the condition evaluates as true. The following is an example of a Repeat Until True loop.



The process performs the following operations:

1. A group of activities executes until the customer records have all been queried. The group consists of:
 - a. A Java Code activity that outputs all the valid customer IDs. When all valid IDs have been output, the activity will output -1 to indicate no more records can be queried.
 - b. A JDBC Query activity that takes each ID and queries a database for the record matching the ID.
 - c. A Send Mail activity that uses the customer information retrieved from the database to send an email to the customer notifying the customer of new product offerings.

For each iteration of the loop, the output of the QueryCustomer activity is placed into a variable named `customerList`.

2. When the condition of the loop evaluates to true, the loop stops executing and transitions to the WriteCustomerList activity so that the customer list will be stored in a file.

The condition evaluates the value of `CustomerID/ID_num`. The Customer ID activity outputs -1 when there are no more customers, so the condition examines the value and when it is -1, the loop can exit.

The following is the configuration for this example Repeat Until True loop:

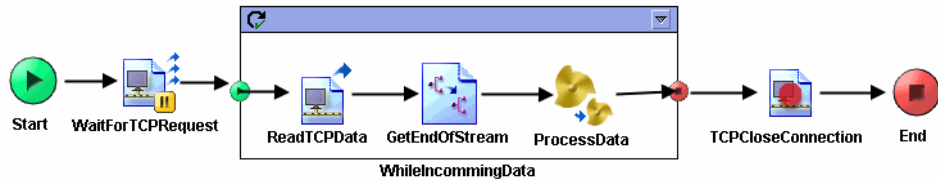
The screenshot shows a 'Configuration' dialog box for a 'Repeat Until True' loop. The dialog has a light blue background and a title bar. It contains several fields and controls:

- Name:** A text field containing 'Group'.
- Description:** An empty text field with a yellow pencil icon to its right.
- Group Action:** A dropdown menu showing 'Repeat-Until-True' with a circular arrow icon.
- Index Name:** An empty text field.
- Conditions:** A text field containing the expression '\$CustomerID/ID_num = -1' with a yellow pencil icon to its right.
- Accumulate output:** A checkbox that is checked.
- Output Activity:** A dropdown menu showing 'QueryCustomer'.
- Output Name:** A text field containing 'custmerList'.

At the bottom right of the dialog are two buttons: 'Apply' and 'Reset'.

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 Wait for TCP Request activity waits for an incoming TCP request.
2. When a request is received, a transition is taken to a While True loop. The output of the GetEndOfStream activity within the group is accumulated in a variable named endOfStream. This allows the condition of the group to be set as follows:

```
not($endOfStream/AccumulatedOutput/EndOfStreamMarker/
    EndOfStreamReached)
```

So, the condition evaluates to true because EndOfStreamReached is false the first time the loop is executed.

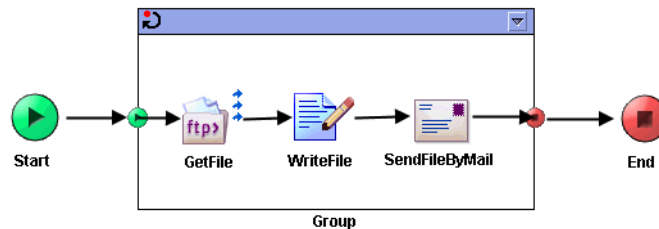
3. Within the group, a Read TCP Data activity reads from the TCP input stream until a separator is reached.
4. The GetEndOfStream activity retrieves the value of \$ReadTCPData/ActivityOutput/endOfStreamReached. This output item of the Read TCP Data activity is set to true when the end of the output stream is reached. So, the GetEndOfStream activity will have an output value of false, until the Read TCP Data activity detects the end of the output stream.
5. The Process Data process is called to handle the incoming data from the TCP stream.
6. The loop repeats until the Read TCP Data activity detects the end of the input stream. When this is detected, the loop condition evaluates to false, and the loop exits.

Repeat On Error Until True Loop

The Repeat On Error Until True loop allows you to repeat a series of activities when an unhandled error occurs. The activities in the group are executed once. If there are no unhandled errors, the loop terminates. If an error occurs for which there is no error transition, the condition of the loop is evaluated — if the condition is true, the loop terminates, if the condition is false, the loop repeats until there is no error occurs or the condition is true.

For example, you may wish to execute a series of activities and retry the execution in the event of an unhandled error. However, you wish to retry the execution only five times, to avoid an infinite loop if the error occurs repeatedly. In this case, specify a repeat on error until true loop with a condition that terminates the execution after five tries.

The following illustrates a repeat on error until true loop.



The process performs the following operations:

1. An FTP Get activity retrieves a file from an FTP server.
2. The contents of the file are written so the data is available at a later time.
3. The contents of the file are sent by email.

The following is the configuration for this example loop:

The screenshot shows a 'Configuration' dialog box with the following fields and values:

- Name:** Group
- Description:** (empty)
- Group Action:** Repeat-On-Error-Until-True
- Index Name:** (empty)
- Conditions:** \$i = 5
- Suspend (If Still Error):** ☐

Buttons at the bottom right: Apply, Reset.

The condition is specified as `$i = 5`, which means that when the index variable is equal to five (that is, the fifth iteration of the loop), the loop exits. The condition is only evaluated upon encountering an unhandled error in the group. If an error is encountered, the loop terminates if the condition evaluates to true. The resulting behavior in this example is that the group of activities executes and loops until either a successful completion of all activities, or until the group is executed five times.

Suspend If Still Error Option

Repeat On Error Until True loops have the Suspend If Still Error option. When this option is checked, the process instance suspends if the error still exists when the condition of the loop is true. The suspended process is displayed by TIBCO Administrator, and the deployment configuration allows you to specify an action to perform if the process is suspended.

This option allows the administrator to correct the problem causing the error. For example, a machine may be down, or an adapter service may not have been started on the machine. After the problem is corrected, the administrator can resume the process execution.

When a process instance resumes execution, the execution resumes before the Repeat On Error Until True loop. The process instance executes the loop, and all process variables are reset to their values before the loop was executed the first time. Therefore, the process instance resumes execution as if the Repeat On Error Until True loop had never executed. The loop is entered again, and if an error occurs, the loop condition is checked as it normally would be.

If the error persists, the process instance continues to be suspended. The administrator can decide whether to resume or kill the process if the error cannot be fixed.

For more information about deployment configuration and specifying actions to perform if processes are suspended, as well as information about how to view suspended processes and resume or kill them, see *TIBCO ActiveMatrix BusinessWorks Administration*.

Chapter 7

Working With Variables

This chapter describes the various types of variables available in TIBCO ActiveMatrix BusinessWorks and how to use these variables in your process definitions.

Topics

- [Overview of Variables, page 106](#)
- [Global Variables, page 107](#)
- [Process Variables, page 111](#)
- [Shared Variables, page 114](#)

Overview of Variables

There are several types of variables in TIBCO ActiveMatrix BusinessWorks, each with their own purpose and usage. TIBCO ActiveMatrix BusinessWorks provides the following types of variables:

- **Global Variables** — these variables allow you to specify constants that can be used throughout the project. The constants can be specified and changed while designing and testing your project. You can also specify different values for each deployment of your project.
- **Process Variables** — these variables allow you to access various data in your project. For example, there are predefined process variables containing the process ID, project name, and other information. You can also create user-defined process variables for containing process-specific data.
- **Shared Variables** — these variables allow you to specify data for use across multiple process instances. Because multiple process instances can access the same variable, you can also synchronize access across processes when setting or retrieving the shared variable.

The following sections describe the types of variables available in your project in greater detail.

Global Variables

Global variables provide an easy way to set defaults for use throughout your project. There are several ways in which they can be used:

- Define a variable using TIBCO Designer, then override the value for individual applications at deployment time using TIBCO Administrator. You can also override values for predefined variables, unless the GUI does not allow you to make them settable later.
- Predefine a variable using TIBCO Designer, then override the value for individual services (for example, publication service or TIBCO ActiveMatrix BusinessWorks process) at deployment time using TIBCO Administrator. The values you specify are then used at runtime.

You can also override values for predefined variables, unless the GUI does not allow you to make them settable later.

- Predefine the variable using TIBCO Designer, then override it on the command line.

For example, you could assign the value 7474 to the predefined global variable `RvDaemon`. You can then use the variable in different sessions in your adapter. If you wish to change the TIBCO Rendezvous daemon for your adapter, you can globally set it to a different value or override it from the command line.

When you want to use the global variable in the fields of a resource, enter the variable name surrounded by `%%` on both sides. Some fields in the configuration panel, such as user name and password fields, allow you to drag and drop global variables into the field.

When the project is deployed and the configured components run, all occurrences of the global variable name are replaced with the global variable value. For example, a global variable named `RvServiceTest` with a value of 7800 would be replaced with 7800.

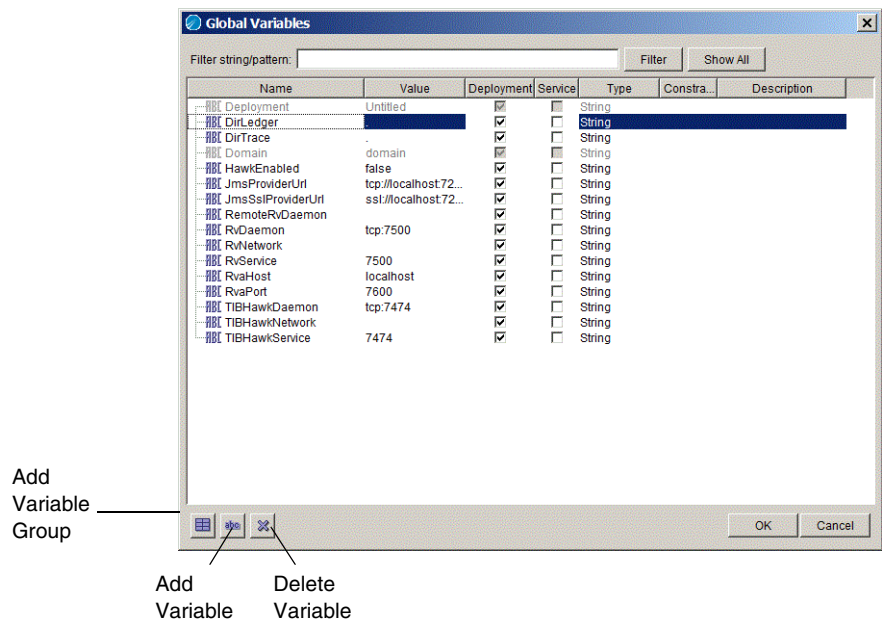
Global Variables Editor

Use the global variables editor to create or modify global variables, mark variables as settable from TIBCO Administrator, and assign a type to a variable.

To display the global variables editor, in the project panel, select the **Global Variables** tab or drop down, then click the Open Advanced Editor (pen) icon. If you select a global variable, then click the pen icon, the editor opens with the variable selected.

[Figure 22](#) shows the global variables editor.

Figure 22 The global variable editor



Adding a Global Variable or Global Variable Group

To add a global variable, select the create variable icon. A new line for the variable appears. Type a name for the variable by triple-clicking the name field.



The name MessageEncoding is reserved and cannot be used to name a global variable. A global variable of this name is created by Designer when generating an EAR file.

When creating a global variable group or variable, it is added to the bottom of the list. When you reopen the global variables editor, the group or variable displays alphabetically in the list.

Global variable groups are used for organizing variables. Variable groups are especially useful if multiple developers share a project using a version control system. To add a global variable group, select the add variable group icon.



You must add at least one variable to a group, or the group will not be saved. If you delete all global variables in a global variable group, the group itself is also automatically deleted.

Using a Filter

By default, all global variables are visible in the editor. You can provide a filter in the **Filter** string/pattern field to limit the display. The filter uses regular expression matching. That is, if you enter **Rv** in the field, only variables that have **Rv** in their name are displayed. Matching is case insensitive. Entering a new expression and clicking **Filter** again performs a new search on all variables and will not refine the current search.

To clear the filter expression and display all variables either click **Show all** or remove the expression from the entry box and click **Filter** again.

Global Variable Attributes

To add or edit a name, value, constraint or description attribute, triple-click in the attribute field. The type attribute has a drop down menu that displays choices. Click in the type field to display the menu.

- **Name** — Provide a name for the variable.
- **Value** — Provide a value for the variable, depending on the type you select.
- **Deployment** — Select the deployment check box to make the variable visible and settable when deploying using TIBCO Administrator. If the check box is clear, the variable is not visible in TIBCO Administrator. Make certain that all variables used in TIBCO ActiveMatrix BusinessWorks process definition have this field checked.
- **Service** — Indicates that the variable should be included when the **Include all service level global variables** option is selected when building the enterprise archive file. A variable can be set on a per-service basis. This option is used for TIBCO adapter archives and TIBCO ActiveMatrix BusinessWorks.
- **Type** — Click in the field to select the variable type, String, Integer, Boolean, or Password. If Password is selected, the value you provide is obfuscated in the repository.
- **Constraint** — For String and Integer types, provide a range of allowed values. The constraint field for Strings is an enumeration, for example, **one, two, three**. The constraint field for Integers is for a range, for example, **1-100**. Note that constraints are currently not implemented in TIBCO Administrator.
- **Description** — Provide a description of the variable.

Deleting a Global Variable

Delete a global variable by selecting it and clicking the delete variable icon. You cannot delete a global variable that is in use. To find where global variables are used, click **Tools > Find Global Variable Usages**.

Changing Global Variable Values at Runtime

You can change the value of a global variable when you deploy your project in TIBCO Administrator. See the section on modifying runtime variables in *TIBCO ActiveMatrix BusinessWorks Administration* for more information on using TIBCO Administrator.

You can also specify values for global variables when starting a process engine on the command line. To do this, specify the following as a command line argument when starting the process engine:

```
-tibco.clientVar.<variablePathAndName> <value>
```

where *variablePathAndName* is the name of the variable you wish to set, including the path to the variable if it is contained in a folder. *value* is the value you wish to set the variable to. For example, if you have a global variable named `item1` contained in a folder named `myGroup` and you wish to set its value to 500, add the following argument to the command line when starting the process engine:

```
-tibco.clientVar.myGroup/item1 500
```

Process Variables

Process variables are data structures available to the activities in the process. Process variables are displayed in the Process Data panel of each activity's Input tab. This allows you to use process data to supply input values for an activity. Each process variable name starts with a dollar sign (\$).

There are four types of process variables:

- [Activity Output](#)
- [Predefined Process Variables](#)
- [Error Process Variables](#)
- [User-Defined Process Variables](#)

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 (with a dollar sign placed in front of the name to indicate it is a process variable).

Activities can use output from previously executed activities by mapping data from the process variable to the activity's input. See [Input on page 62](#) for more information about the Input tab and mapping process data to an activity's input.

Predefined Process Variables

There are two process variables that are available to all activities that accept input: `$_globalVariables` and `$_processContext`. `$_globalVariables` contains the list of global variables defined on the Global Variables tab of the project. See [Global Variables on page 107](#) for more information about global variables. `$_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. However, global variables of the type 'Password' are not listed under the `$_globalVariables` process variable.

Error Process Variables

When an error occurs in a process, the data pertaining to the error is placed into process variables. The `$_error` process variable contains general error information. Activities can also have error process variables named `$_error_<activityName>`. In the event of an error, the activity's error variable is populated with the appropriate error schema.

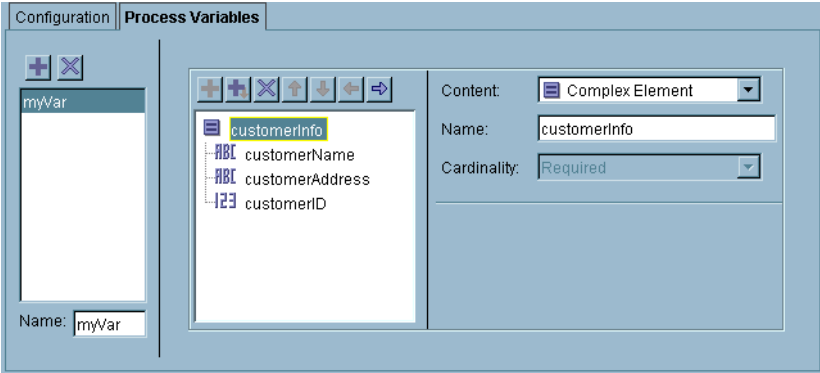
See [Chapter 10, Error Handling, on page 174](#) for more information about handling errors and error process variables.

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 Process Definition resource. You create a process variable in the same way you create data schemas for activities. See [Editor on page 56](#) for more information about creating data schemas.

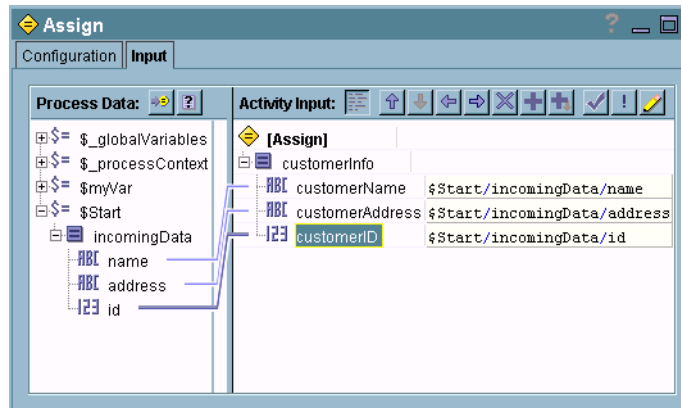
[Figure 23](#) illustrates creating a process variable. You add a process variable and give it a name in the left-most panel, then you create its schema in the middle panel.

Figure 23 Creating a process variable



To assign a value to a user-defined process variable, use the Assign activity. Assign allows you to specify which process variable you wish to modify. Only user-defined process variables can be modified. You can then specify a new value for the process variable in the Input tab of the Assign activity. [Figure 24](#) illustrates assigning a value to a user-defined process variable.

Figure 24 Assigning a value to a user-defined process variable



When you use the Assign activity, the entire contents of the process variable is replaced by the specified new contents. That is, if you do not supply a value for an element in a process variable, that element no longer contains a value at run time. Therefore, use the Assign activity to reset the value of a process variable, and make sure to supply the correct values for all elements in the process variable.

See *TIBCO ActiveMatrix BusinessWorks Palette Reference* for more information about the Assign activity.

Memory Usage of Process Variables

All process variables in a running process instance are stored in memory and therefore consume system resources. Memory saving mode allows memory used by a process variable to be released when the value of the variable is no longer needed. When using memory saving mode, as each activity is executed, the list of process variables is evaluated to determine if subsequent activities in the process refer to the variable. If no activities refer to the variable, the memory used by the variable is released.

Memory saving mode can reduce the memory used by actively running process instances, as well as potentially improve the performance of checkpoints. By default, memory saving mode is disabled, but you can enable memory saving mode for specific process instances by setting the `EnableMemorySavingMode.<processname>` property to true. You can enable memory saving mode for all process instances by setting the `EnableMemorySavingMode` property to true.

See *TIBCO ActiveMatrix BusinessWorks Administration* for more information about setting properties.

Shared Variables

A shared variable is a shared configuration resource in the General Activities palette. There are two types of shared variables:

- [Shared Variable](#)
- [Job Shared Variable](#)

Shared Variable

A Shared Variable resource allows you to share data across process instances. All process instances can read and update the data stored in a shared variable. This type of shared variable is useful if you wish to pass data across process instances or if you wish to make a common set of information available to all process instances. For example, you may have a set of approval codes for incoming orders that change daily for security purposes. You can create a shared variable to hold the approval codes and create one process definition for setting the codes. You can then retrieve the shared variable in all processes that require the current approval codes.

Job Shared Variable

A Job Shared Variable resource is similar to a Shared Variable, but its scope is limited to the current job. A copy of the variable is created for each new process instance. This type of shared variable is useful for passing data to and from sub-processes without creating an input or output schema for the called process.

You can use the Get Shared Variable and Set Shared Variable activities to access the data instead of mapping data to a called processes input or output schemas. New process instances receive a copy of the Job Shared Variable, so data cannot be shared across process instances. Therefore, if a called process has the Spawn configuration field checked, a new process instance is created and the process instance receives a new copy of the Job Shared Variable.

Configuring Shared Variables

You configure a shared variable by providing the following:

- a name for the variable
- the schema of the data contained in the variable
- optionally, an initial value for the variable

- for Shared Variable resources, whether the variable is persistent and whether the variable should be available across process engines

Setting the Schema of the Shared Variable

The Schema tab allows you to specify a complex schema for shared variable data. See [Editor on page 56](#) for more information about specifying data schema.

Setting the Initial Value of the Variable

While configuring the shared variable, you can specify an initial value for the variable on the Initial Value tab. You can choose either None (no initial value), Select Value, or Build Value.

When you choose None, no initial value is set and the variable value must be set in a process definition before it can be retrieved. When you choose Select Value, you can choose a stored XML Instance resource containing data that matches the specified schema. When you choose Build Value, an Edit button appears that allows you to bring up a dialog to construct the initial value of the variable.

Making the Variable Persistent

The current value of the shared variable is stored in memory for efficient access by process instances. However, if a process engine crashes, the current state of the shared variable may be lost. To preserve the current state of a Shared Variable resource, you can check the Persistent field on the Configuration tab.

When a shared variable is persistent, its current state is written to the process engine storage location (either the file system or a database depending upon how the process engine was deployed). The current state of the shared variable is only updated in the process engine persistent storage when the value of the variable changes.



The Persistent field is not available on Job Shared Variable resources, but the state of these variables is saved by a checkpoint. Therefore, it is not necessary to store these variables in a separate location in the process engine storage.

Sharing the Variable Across Multiple Process Engines

If you wish to make the value of a Shared Variable resource available to process instances across multiple process engines, you can check the Multi-Engine field on the Configuration tab. This field is not available on Job Shared Variable resources.

If you choose this option, a database must be used to store process engine data. Also, only engines that are in the same deployment and part of the same load-balancing group can share variables. See *TIBCO ActiveMatrix BusinessWorks Administration* for more information on specifying load balancing groups and specifying a database for process engine storage during deployment. See [Loading Processes to Test on page 316](#) for more information about configuring a database for process engine storage within the testing environment.

Because multiple process engines access the shared variable, the current value of the variable is not stored in memory. Both retrieving the current value of the variable and assigning a new value to the variable requires I/O to the process engine database storage.

Assigning and Retrieving the Variable's Value

You can retrieve the current value of a shared variable by using the Get Shared Variable activity. You can assign a value to the shared variable by using the Set Shared Variable activity. Both of these activities can be found in the General Activities palette.

Before you retrieve the value of a shared variable, you must either set an initial value in the variable's configuration or use the Set Shared Variable activity to set the variable value. You can also configure the Set Shared Variable activity to include the current value of the variable as output for the activity. This provides access to the variable's current value in subsequent activities in the process definition.

Synchronizing Access to Shared Variables

Because multiple process instances can potentially access and assign values to Shared Variable resources, the Lock shared configuration object and critical section group allow you to synchronize access to Shared Variable resources. Without a mechanism for locking, a process instance could update the value of a variable while another process instance is attempting to read the value. This would result in an unpredictable value for the variable.

You should use critical section groups to contain the Set Shared Variable and Get Shared Variable activities. This ensures that only one process instance attempts to assign a value to the variable and ensures that no process assigns a value to the variable when the current process attempts to read the value.

See [Critical Section Groups on page 90](#) for more information about using the critical section group.

Chapter 8

Mapping and Transforming Data

This chapter describes mapping data from a process to a specific activity's input and how to transform the data using the activity's Input tab.

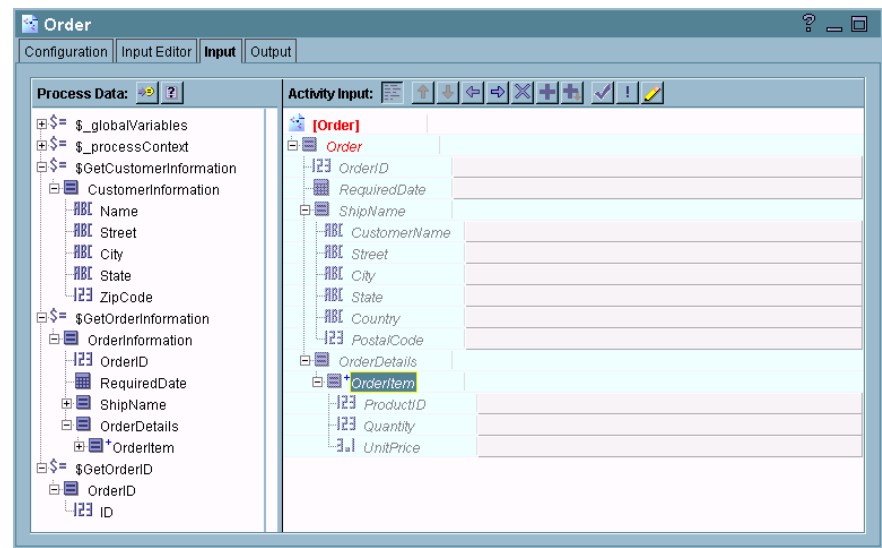
Topics

- [Overview of Mapping and Transformation, page 118](#)
- [Buttons, Menus, and Icons, page 121](#)
- [Specifying Constants, page 126](#)
- [Data Validation, page 127](#)
- [Repairing Incorrect Mappings, page 130](#)
- [Shortcuts, page 132](#)
- [Examples of Mappings, page 137](#)
- [XSLT Statements, page 155](#)

Overview of Mapping and Transformation

The Input tab of an activity allows you to supply the data that an activity expects as input. On the Input tab, you can see the available process data and the activity’s expected input. The process data and activity input are represented as schema trees. [Figure 25](#) illustrates an activity’s Input tab.

Figure 25 An activity’s input tab



Process Data Panel

The process data is the list of available data in the process definition at the point where the activity is located. An activity has access to all output data from any activity that has been previously executed in the process definition. All activities also have access to global variables, process variables, and information about the current process context. You cannot modify the process data on an activity’s input tab, but you can use the data to supply input to the activity.

Activity Input Panel

The activity's input is an Extensible Stylesheet Language Transformation (XSLT) template that specifies how process data should be transformed to provide the expected input. Normally, you do not need detailed knowledge of XSLT to specify an activity's input. However, if you are familiar with XSLT and you wish to see the actual code, you can right-click on any item in the input schema and choose Copy from the popup menu. Then open a blank text document and choose Paste. The XSLT is displayed in your text document.

You can also use your own XSLT templates to perform transformations instead of using the techniques described in this chapter. You can paste XSLT into your activity input or use the XSLT File shared configuration resource and the Transform XML activity. See *TIBCO ActiveMatrix BusinessWorks Palette Reference* for more information and examples of using XSLT to perform mapping.

Mapping and Transforming Process Data to Activity Input

When an activity is first dragged from a palette to the design panel, the activity's input elements are displayed as hints. These hints show you the data the activity expects as input. Each element can be required or optional. Required elements must have a mapping or formula specified. Once a mapping or formula is specified, a hint becomes an XSLT statement. See [Statements, Hints, and Errors on page 120](#) for more information about hints and statements.

You map data by selecting an item in the Process Data panel, then drag and drop that item into the desired schema element you wish to map in the Activity Input panel. When you perform mapping, simple mappings appear in the formula area next to the input element after you release the mouse button. For more complex mappings, the Mapping Wizard dialog allows you to select which kind of mapping you wish to perform.

Most options in the Mapping Wizard dialog are straightforward. However, there are some complex scenarios that require multiple steps. Many of these situations are described in the section [Examples of Mappings on page 137](#). You may also wish to refer to [XSLT Statements on page 155](#) for a reference of XSLT statements when deciding which XSLT statement can be used to achieve the result you desire.

You can specify XPath formulas to transform an element if you need to perform more complex processing. The XPath Formula Builder allows you to easily create XPath formulas. For more advanced use of XPath, see [Chapter 9, XPath, on page 161](#). There are also a variety of third-party books and resources about XSLT and XPath.

Statements, Hints, and Errors

When you display the Input tab for an activity, the existing statements are examined, and any input elements that do not have a statement are displayed as hints. Hints are reminders that you can specify a statement for the input element, but they are not stored as part of the XSLT template for the activity's input. Hints are displayed in italics with a light blue background. Once you specify a mapping or a formula for a hint, the input element becomes a statement. You can also drag the hint to the left past the dividing line between the panels and the hint becomes a blank statement.

Once you specify a statement in the Activity Input panel and apply the change to the activity, it becomes part of the XSLT template used to create the activity's input data. These statements are saved as part of the activity in the repository. Statements are only deleted if you manually delete them using the delete button, or if you use the Mapper Check and Repair button to automatically fix errors. Therefore, if the input schema for the activity changes, your statements may no longer be valid. See [Repairing Incorrect Mappings on page 130](#) for more information about using the Mapper Check and Repair button to fix statements in the activity's input.

Any statement or hint that has an error is displayed in red. A hint is only displayed in red if it is a required input element. All required input elements must have statements specified. If a statement is red, you must fix the error before executing this process definition. The Mapper Check and Repair button can help you automatically fix some errors. See [Repairing Incorrect Mappings on page 130](#) for more information about fixing errors.

Buttons, Menus, and Icons

The Input tab contains several toolbar buttons, popup menus, and icons. This section describes the various graphical elements of the Input tab.

Toolbar and Right-Click Menu on the Input Tab

The Process Data panel and the Activity Input panel have several buttons for performing various functions. There is also a popup menu when you right-click on elements in each panel. [Table 11](#) describes the buttons and right-click menu items available in the panels of the Input tab.

Table 11 Input tab toolbar buttons

Button	Right-Click Menu	Description
Process Data Area		
		Coercions. Allows you to specify a type for Process Data elements that are not a specific datatype. For example, a choice element can be coerced into one of the possible datatypes for the element, or an element of datatype any can be coerced into a specific datatype.
		Type Documentation. Allows you to specify or view documentation for schema elements.
	Expand	This menu item has two sub-menus: Content and All. Expand > Content expands the current element so that all elements that are currently used in a mapping are visible. Expand > All expands all sub-elements of the currently selected element.
	Show Connected	Expands the elements in the Activity Input area to display elements that are mapped to the currently selected element or its sub-elements.
	Delete	Deletes the selected element.
	Copy	Copies the selected element. The element can be later pasted.
Activity Input Area		
		Shows/hides the mapping formulas for the input elements.

Table 11 Input tab toolbar buttons

Button	Right-Click Menu	Description
		Move Up. Moves the selected element up in the activity input tree.
		Move Down. Moves the selected element down in the activity input tree.
	Move Out	Move Out. Promotes the selected element to the next highest level in the activity input tree.
	Move In	Moves the currently selected element into a new statement. This displays the Move Into New Statement dialog that allows you to choose the statement you wish to move the element into. See XSLT Statements on page 155 for more information about XPath statements.
	Delete Delete All	Deletes the selected element and all of its child elements.
		Insert. Inserts a new element or XSLT statement in the activity input schema on the same level of the hierarchy as the currently selected element. You can add one element or XSLT statement at a time with this button. The right-click menu item Statement provides a shortcut for multi-line statements, such as Choice or If. See the description of the Statement menu item below for more information. See XSLT Statements on page 155 for more information about XSLT statements.
		Add Child. Adds a statement for a child element to the currently selected element.
		Mapper Check and Repair. Verifies the XSLT template you have created in the Activity Input panel against the activity's expected input. A list of errors and warnings appear and you can choose which items you wish to fix. TIBCO ActiveMatrix BusinessWorks attempts to fix simple problems such as adding missing activity input items that are expected. See Repairing Incorrect Mappings on page 130 for more information.

Table 11 Input tab toolbar buttons

Button	Right-Click Menu	Description
		Edit Statement. Allows you to modify an XSLT statement for the element. See XSLT Statements on page 155 for more information about XSLT statements.
	Edit	XPath Formula Builder. Invokes the XPath formula builder. You can use this editor to create an XPath statement for this input element. See Chapter 9, XPath, on page 161 for more information about XPath and the XPath formula builder.
	Expand	This menu item has three sub-menus: Content, Errors, and All. Expand > Content expands the current element so that all sub-elements that have a mapping or expression are visible. Expand > Errors expands the currently element so that all sub-elements that have an error in their expression are visible. Expand > All expands all sub-elements of the currently selected element.
	Show Connected	Expands the elements in the Process Data area to display elements that are mapped to the currently selected element or its sub-elements.
	Statement	This menu item contains shortcuts that allow you to easily add the desired XSLT statement(s) with one menu item instead of adding the statement(s) with the Insert button. See Right-Click Menu on page 132 for a description of the sub-items of this menu.
	Undo	Rolls back the last operation performed.
	Redo	Performs the last operation that was undone with the Undo menu item.
	Cut	Deletes the selected element. The element can be later pasted to a new location.
	Copy	Copies the selected element.
	Paste	Pastes the last element that was copied or cut.

Icons for Schema Element Datatypes

Schema elements also have a set of associated icons to indicate their type. [Table 12](#) describes the icons used for schema items.



You can use the Type Documentation button to obtain any available documentation on any node in the Process Data or Activity Input schema trees.

Table 12 Icons for schema items

Icon	Description
	Complex element that is a container for other datatypes. This is also called a <i>branch</i> in the schema tree.
	Simple string or character value.
	Simple integer value.
	Simple decimal (floating point) number.
	Simple boolean value.
	Simple Date or Time. This can be any of the following datatypes: • Time • Date • Date & Time • Duration • Day • Month • Year • Month & Year • Day & Month
	Simple binary (base 64) or hex binary value.
	Represents a schema item that can be any datatype. Data in this schema element can be any datatype.
	Choice. Specifies that the actual schema element can be one of a specified set of datatypes.

Qualifier Icons

Schema elements can have additional icons that specify qualifications. The qualifier icons have different meanings depending upon where they appear. For example, a question mark icon signifies an element is optional in the Process Data schema or in a hint in the Activity Input. However, in an XSLT statement, the question mark signifies the statement is "collapsed" and an implicit "if" statement is set, but not displayed in the Activity Input area.

[Table 13](#) describes the additional qualifiers that appear next to the name of schema items.

Table 13 Additional icons for hints

Qualifier	Process Data or Hint	Statement
	No qualifier indicates the element is required.	N/A
	A question mark indicates an optional Item.	An implicit "if" statement is set for this statement. This occurs when you map an optional element from the process data to an optional element in the Activity Input schema or if you specify Surround element with if test on the Content tab of the Edit Statement dialog.
	An asterisk indicates the item repeats zero or more times.	N/A
	A plus sign indicates the item repeats one or more times.	N/A
	A null sign indicates the item may be set to null.	A null sign indicates the item is explicitly set to null. You can set an element explicitly to null by clicking the Edit Statement button for the element, then checking the Set Explicit Nil field on the Content tab of the Edit Statement dialog.

Specifying Constants

For each element in the activity input schema tree, you can specify a constant. Constants can be strings or numeric values. To specify a string, enclose the string in quotes. To specify a number, type the number into the schema element's mapping field. The following illustrates specifying the string "USA" for the Country item and 94304 for the PostalCode item of an input schema.



Constants can also be used in functions and search predicates. To learn more about complex XPath expressions that use functions and search predicates, see [Chapter 9, XPath, on page 161](#).

Date and Datetime Strings in Constants

In constant expressions used in activity input bindings, datetime values are read in according to the ISO 8601 standard, as described in the XML Schema specification. For example, the value:

"2002-02-10T14:55:31.112-08:00"

is 55 minutes, 31 seconds and 112 milliseconds after 2pm on February 10th, 2002 in a timezone that is 8 hours, 0 minutes behind UTC.

If no timezone field is present, the value is interpreted in the timezone of the machine that is performing the parsing. This can lead to complications if you are processing data from different timezones, so you are encouraged to always use timezones.

When TIBCO ActiveMatrix BusinessWorks generates datetime strings (for example in the process debugger display for process data), UTC time is always used. The output for the example above is:

2002-02-10T14:55:31.112Z

which is the equivalent time in the UTC timezone.

Data Validation

Data passed as input to an activity or from an event received by a process starter is validated to ensure that it conforms to its specified datatype. For example, if you pass an XML document to a Parse XML activity, the content of the document is checked to ensure that elements in the document are actually of the specified datatypes.

Table 14 describes the validation behavior. Datatype validation listed with the prefix `xsd:` is defined in the namespace <http://www.w3.org/2001/XMLSchema>. See *XML Schema Part2: Datatypes* specification at <http://www.w3.org/TR/2004/PER-xmlschema-2-20040318/> for more information on the proper representation of these datatypes. Datatype validation listed with the prefix `xdt:` is defined in the namespace <http://www.w3.org/2003/11/xpath-datatypes>. See *Xquery 1.0 and Xpath 2.0 Functions and Operators* specification at <http://www.w3.org/TR/2003/WD-xpath-functions-20031112/> for more information on the proper representation of these datatypes.

Table 14 Datatype validation

Data Type	Validation
Built-In Primitive Types	
boolean	xsd:boolean
decimal	xsd:decimal
float	xsd:float
double	xsd:double
string	xsd:string
duration	xsd:duration
yearMonthDuration	xdt:yearMonthDuration
dayTimeDuration	xdt:dayTimeDuration
dateTime	xsd:dateTime
time	xsd:time
date	xsd:date
gYearMonth	xsd:gYearMonth

Table 14 Datatype validation

Data Type	Validation
gYear	xsd:gYear
gMonthDay	xsd:gMonthDay
gDay	xsd:gDay
gMonth	xsd:gMonth
hexBinary	xsd:hexBinary
base64Binary	xsd:base64Binary
anyURI	xsd:anyURI
QName	xsd:QName
NOTATION	xsd:NOTATION
untypedAtomic	xdt:untypedAtomic
Built-In Derived (Atomic) Types	
integer	xsd:integer
nonPositiveInteger	xsd:nonPositiveInteger
negativeInteger	xsd:negativeInteger
long	xsd:long
int	xsd:int
short	xsd:short
byte	xsd:byte
nonNegativeInteger	xsd:nonNegativeInteger
unsignedLong	xsd:unsignedLong
unsignedInt	xsd:unsignedInt
unsignedShort	xsd:unsignedShort
unsignedByte	xsd:unsignedByte
positiveInteger	xsd:positiveInteger
normalizedString	xsd:normalizedString

Table 14 Datatype validation

Data Type	Validation
token	xsd:token
language	xsd:language
Name	xsd:Name
NCName	xsd:NCName
ID	xsd:ID
IDREF	xsd:IDREF
ENTITY	xsd:ENTITY
NMTOKEN	xsd:NMTOKEN

Repairing Incorrect Mappings

Any incorrect statements are displayed in red in the Activity Input panel. Errors can occur for a number of reasons. For example,

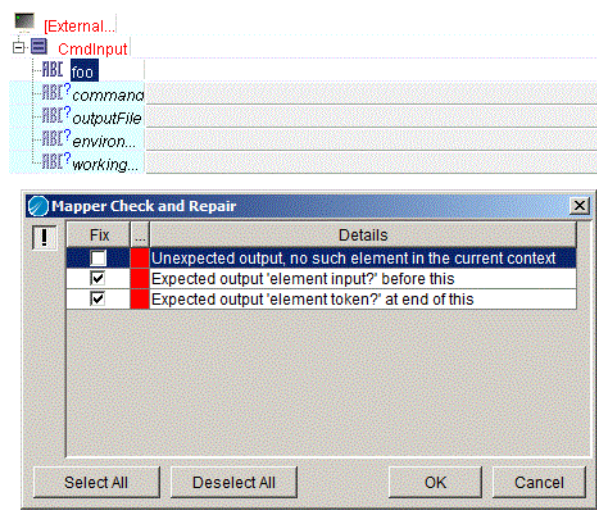
- a required element has no statement, and therefore must be specified
- the activity’s input schema has changed and existing statements may no longer be valid
- the XPath formula for an element may contain an error

You should correct any errors before attempting to test or deploy your process definition.

To help find potential problems in your mappings, click the Mapper Check and Repair button. This button displays a dialog with all potential problems in the specified mappings. You can select the Fix checkbox for potential errors, and TIBCO ActiveMatrix BusinessWorks will attempt to fix the problem.

For example, [Figure 26](#) illustrates an activity input schema for the External Command activity. A new element named foo has been added to the schema and the expected element token has been removed. The Mapper Check and Repair dialog displays these problems with the Fix checkbox checked. You can select or clear the Fix checkbox for each item, depending upon whether you wish to fix the problem or leave the problem until a later time. When you click OK, any items that are marked for fixing are repaired.

Figure 26 Mapper check and repair dialog



Some potential problems in the Mapper Check or Repair dialog cannot be fixed easily, and therefore there is no checkbox in the Fix column for these items. For example, if an element expects a string and you supply a complex type, the corrective action to fix the problem is not clear, and therefore TIBCO ActiveMatrix BusinessWorks cannot automatically fix the problem. You must repair these items manually.

If you want to return to the original expected activity input and remove all of the currently specified mappings, perform the following:

1. Delete the root element of the activity's input by selecting it and clicking the Delete button.
2. Click the Mapper Check and Repair button.
3. Select the Fix checkbox for all items.
4. Click OK.

Alternatively, you can simply select the root input element and press the delete key on your keyboard as a shortcut for the procedure above.

After deleting all mappings and schema items and then repairing the input schema, the activity's input reverts to the state when the activity is first dragged into the design panel.

Shortcuts

The Move In, Insert, Add Child, and Edit Statement buttons on the Input tab toolbar are ways to manually manipulate XSLT statements in the Activity Input area. These buttons, however, only add or modify one statement at a time. Also, there are some situations where you wish to convert a hint into a statement without performing any mapping. This section describes shortcuts for manipulating XSLT statements.

Right-Click Menu

When you select an element in the Activity Input schema and right-click, a popup menu appears. The Statement menu item contains several sub-items that are useful shortcuts for creating XSLT statements.

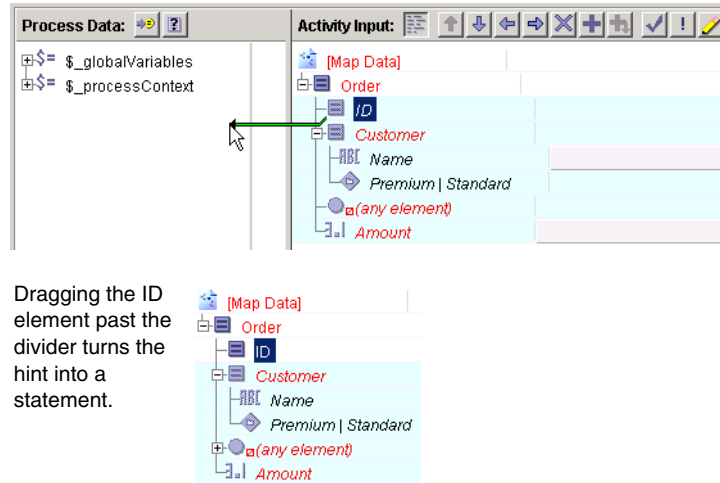
- Surround with Choice — a shortcut for adding a choice statement and its associated conditions or otherwise statements around the currently selected element.
- Surround with If — a shortcut for adding an if statement and placing the currently select element as the sub-element of the if.
- Surround with For-Each — a shortcut for moving the current element into a For-Each statement.
- Surround with For-Each-Group — a shortcut for moving the current element into a For-Each-Group statement and adding a Group-By grouping statement.
- Duplicate — a shortcut for creating a duplicate of the currently selected element (including any mappings or XPath formulas for the element). The duplicate is added below the currently selected element.
- Insert Model Group Content — a shortcut for inserting the contents of a selected model group into the mapper tree. The selected element in the Activity Input Schema is replaced by the contents of the model group you select.

Dragging to the Left

Dragging an element in the Activity Input schema to the left past the divider between the two areas of the Input tab changes a hint into an XSLT statement. [Figure 27](#) illustrates dragging an element to the left.

Figure 27 Dragging to the left to change a hint to a statement

Before dragging, the ID element is a hint.



This shortcut is useful in the following situations:

- When you have a complex element with no sub elements and no content.
- When you have a choice element, dragging to the left brings up the Mapping Wizard and allows you to choose a type for the element.
- When you have an element of type Any, dragging to the left brings up a dialog that allows you to specify the type for the element.

Cutting and Pasting

The Activity Input area is an XSLT template for specifying the activity's input schema. You can choose any element in the Activity Input area and select Copy from the right-click menu or press the Control-C keys to copy the XSLT statement for the element. Once the XSLT is copied, you can paste it into a text editing tool to view or modify the code.

You can also paste arbitrary XSLT code into the Activity Input area using the right-click menu or the Control-V keys. Pasting XSLT code from the copy buffer places the code above the currently selected element in the Activity Input area.

Automatic Testing

When you map process data elements to activity input elements, the behavior of the mapping depends upon the types of elements you are mapping. In the simplest case of mapping a required element in the process data schema to a required activity input element, the value of the process data element is assigned to the required activity input element.

However, when elements are optional or nillable, more complex tests are necessary. When you drag the process data element to the activity input element, the necessary tests are automatically placed into the activity input XSLT template.

This section describes the result of mapping different types of elements. The types of mappings are described, then an example is given that illustrates these mappings and shows the XSLT code that is generated automatically when these mappings are performed.

Required to Required

Specifies that the statement should always include the required activity input element and its value should be obtained from the required process data element that the element is mapped to.

Optional to Optional

Specifies that the statement should test if the process data element is present, and if so, include the optional element in the activity's input. If the process data element is not present, the optional element is omitted from the activity's input.

Nillable to Nillable

Specifies that both the process data and activity input elements can be nil. Therefore, the value of the activity input element is set to the value of the process data element. The value of the activity input element is set explicitly to nil if that is the value of the process data element.

Optional to Nillable

Specifies that the statement should test if the optional process data element exists. If the element exists, the activity input element should be created and set to the value of the process data element. If the process data element does not exist, the element is omitted from the activity input schema.

Nillable to Optional

Specifies that the statement should test if the process data element has a value specified, and if so, the optional element in the activity input should be set to the value of the process data element. Otherwise, if the process data element is nil, the optional element is omitted from the activity input.

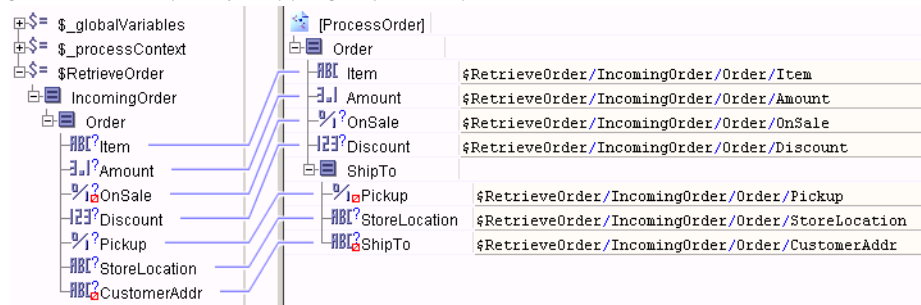
Optional & Nillable to Optional & Nillable

Specifies that if the optional process data element exists, then include the optional activity input element in the input schema. If the process data element is nil, set the value of the activity input element explicitly to nil. If the process data element is not nil, set the value of the activity input element to the value of the process data element. If the process data element is not present, then omit the optional element from the activity input schema.

Example of Mapping Required, Optional, and Nillable Elements

Figure 28 illustrates the different types of mappings when elements are required, optional, or nillable.

Figure 28 Examples of mapping required, optional, and nillable elements



In the example above, Item and Amount illustrate mapping required elements to other required elements. The OnSale element illustrates mapping a nillable element to an optional element. The Discount element illustrates mapping an optional element to an optional element. The Pickup element illustrates mapping an optional element to a nillable element. The CustomerAddr and ShipTo elements illustrate mapping an optional and nillable element to an optional and nillable element.

Below is the XSLT template illustrated by the mappings in Figure 28. You can see from the XSLT code that certain mappings are automatically surrounded by tests to create the appropriate input schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:template xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```

xmlns:pd="http://xmlns.tibco.com/bw/process/2003"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:pfex="http://www.tibco.com/xmlns/ae2xsd/2002/05/ae/Order">
<Order>
  <Item>
    <xsl:value-of select="$RetrieveOrder/IncomingOrder/Order/Item"/>
  </Item>
  <Amount>
    <xsl:value-of select="$RetrieveOrder/IncomingOrder/Order/Amount"/>
  </Amount>
  <xsl:if test="$RetrieveOrder/IncomingOrder/Order/OnSale/@xsi:nil!=
    (&quot;true&quot;;&quot;1&quot;)">
    <OnSale>
      <xsl:value-of select="$RetrieveOrder/IncomingOrder/Order/OnSale"/>
    </OnSale>
  </xsl:if>
  <xsl:if test="$RetrieveOrder/IncomingOrder/Order/Discount">
    <Discount>
      <xsl:value-of select="$RetrieveOrder/IncomingOrder/Order/Discount"/>
    </Discount>
  </xsl:if>
  <pfex:ShipTo>
    <Pickup>
      <xsl:choose>
        <xsl:when test=
          "exists($RetrieveOrder/IncomingOrder/Order/Pickup)">
          <xsl:value-of select=
            "$RetrieveOrder/IncomingOrder/Order/Pickup"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:attribute name="xsi:nil">true</xsl:attribute>
        </xsl:otherwise>
      </xsl:choose>
    </Pickup>
    <xsl:if test="$RetrieveOrder/IncomingOrder/Order/StoreLocation">
      <StoreLocation>
        <xsl:value-of select=
          "$RetrieveOrder/IncomingOrder/Order/StoreLocation"/>
      </StoreLocation>
    </xsl:if>
    <xsl:if test="$RetrieveOrder/IncomingOrder/Order/CustomerAddr">
      <ShipTo>
        <xsl:copy-of select=
          "$RetrieveOrder/IncomingOrder/Order/CustomerAddr/@xsi:nil"/>
        <xsl:value-of select=
          "$RetrieveOrder/IncomingOrder/Order/CustomerAddr"/>
      </ShipTo>
    </xsl:if>
  </pfex:ShipTo>
</Order>
</xsl:template>

```

Examples of Mappings

Some mappings require several steps to achieve the desired results. This section describes some complicated mapping scenarios and how to achieve the desired mappings using the tools available.



There are many methods to insert or modify XSLT statements in the Activity Input schema. The examples in this section illustrate the simplest procedures to obtain the desired results. However, you do not have to follow the same procedures outlined in this section to achieve the correct set of statements.

Using XPath

TIBCO ActiveMatrix BusinessWorks uses XPath (XML Path Language) to specify and process elements of the Activity Input schema. You can also use XPath to perform basic manipulation and comparison of strings, dates, numbers, and booleans.

One of the most common uses of XPath is to filter a list for specific elements. XPath expressions have search predicates for performing filtering. In the Activity Input area, when a search predicate is required, it appears as [*<< Filter >>*] to indicate that you must supply the filter expression. For example, you may wish to select a specific order from a list of orders where the item ordered is itemID #34129. To do this, the XPath expression might be:

```
$RetrieveOrders/Order[itemID=34129].
```

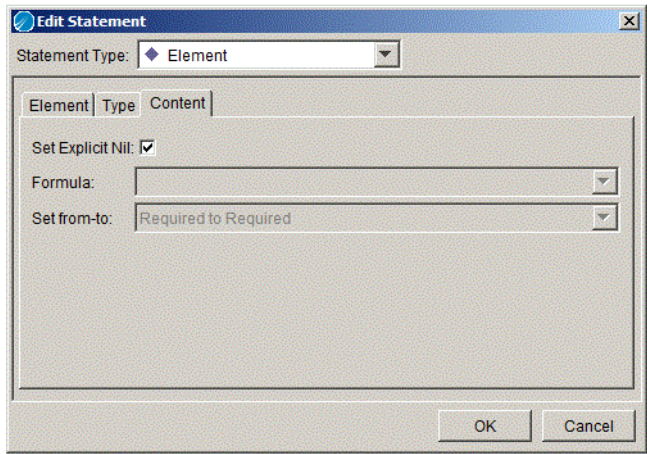
You can use any valid XPath expression in the XSLT statements in the Activity Input area. See [Chapter 9, XPath, on page 161](#) for more information on how to use XPath.

Setting an Element Explicitly to Nil

In some situations, you may wish to set an element explicitly to nil. One situation is when you wish to insert a row into a database table and you wish to supply a NULL for one of the columns. To set an input element explicitly to nil, perform the following:

1. Select the input element you wish to set to nil.
2. Click the Edit Statement button on the Input tab toolbar.
3. Click the Content tab of the Edit Statement dialog.

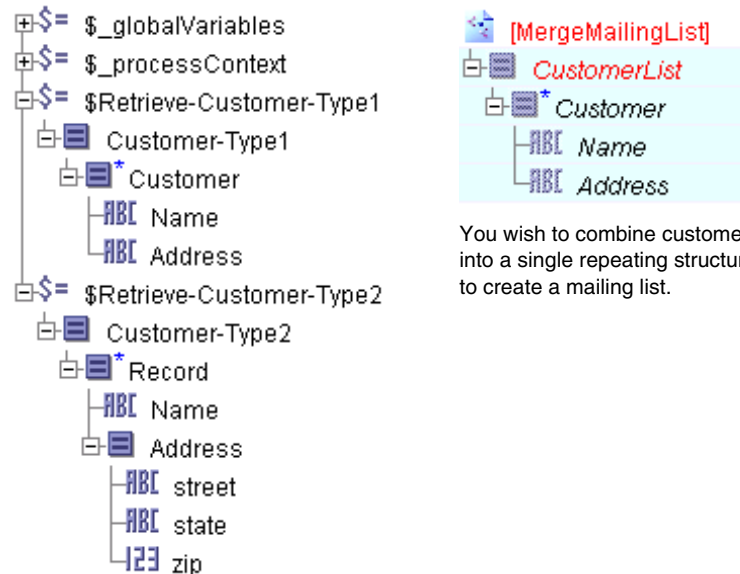
- 4. Check the checkbox in the Set Explicit Nil field.



The element’s formula becomes blank and uneditable (because nil is the value of the element) and the explicit nil qualifier icon appears next to the statement  .

Merging Input from Multiple Sources

You may have multiple elements in the Process Data that you wish to map to one repeating element in the Activity Input. For example, you may have multiple formats for customer records and you wish to create a single, merged mailing list containing all customers in one format. In this example, the schemas are the following:

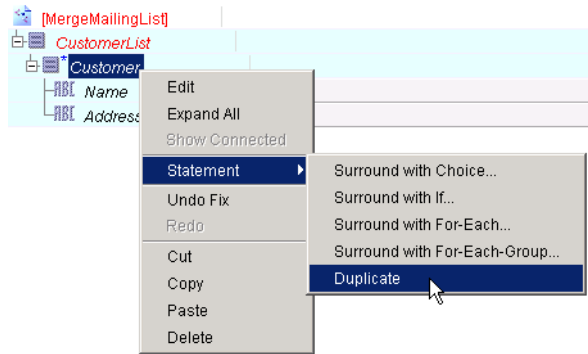


You wish to combine customers into a single repeating structure to create a mailing list.

Multiple types of customers are retrieved, each having a different format for the address.

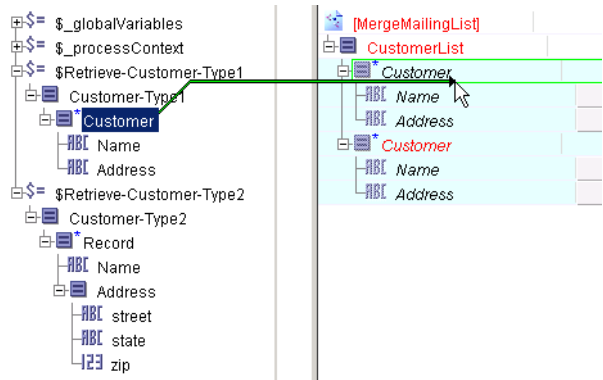
The following procedure describes how to map multiple elements into a single repeating element.

1. Select the repeating element in the Activity Input area, right-click, and select **Statement > Duplicate** from the popup menu.

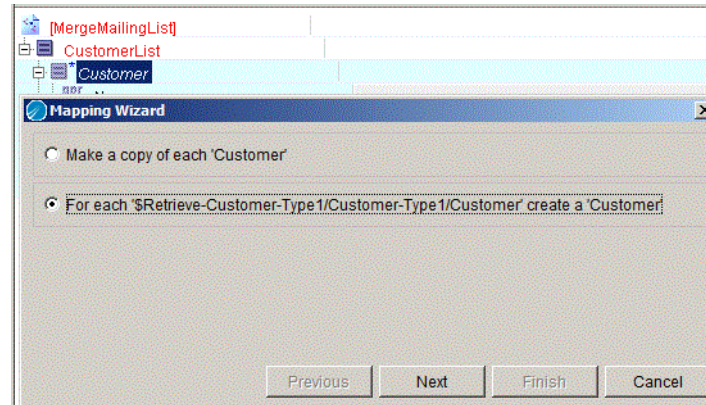


Because you are creating two different formulas for mapping, you need two copies of the repeating element, one for each format. The resulting output contains only one repeating customer element, but the two copies in the Activity Input area make it simpler to perform two different mappings.

2. Map one of the elements from the Process Data to the first copy of the repeating element in the activity input. For example, map `$Retrieve-Customer-Type1/Customer` to `MergeMailingList/CustomerList/Customer`.

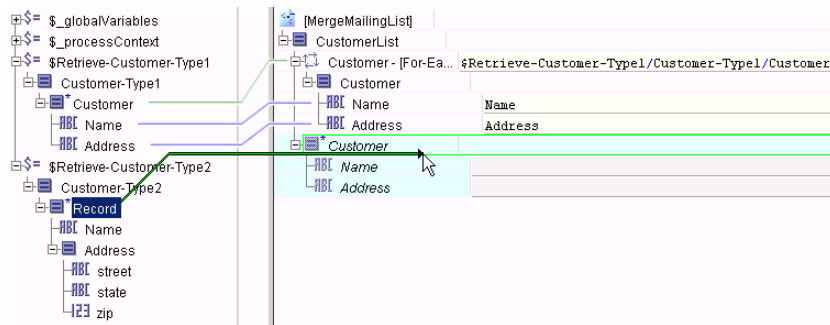


The Mapping Wizard dialog appears and presents choices for what you would like to accomplish. Choose the For Each option and click **Next**.

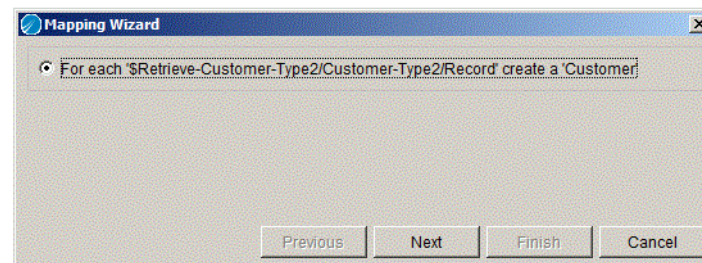


The mapping wizard asks if you wish to automatically map items with the same names. Click **Finish** to accept the default mappings.

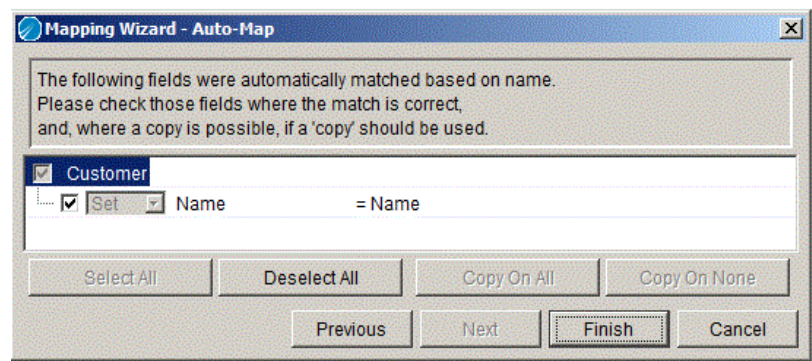
3. Map the other element from the Process Data to the second copy of the repeating element in the activity input. For example, map `$Retrieve-Customer-Type2/Record` to `MergeMailingList/CustomerList/Customer`.



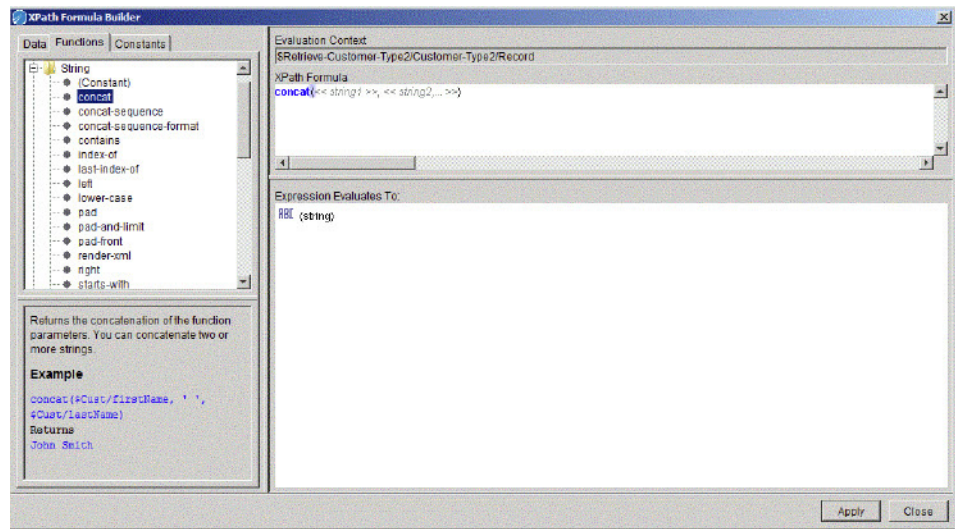
In the Mapping Wizard dialog, choose the For Each option and click **Next**.



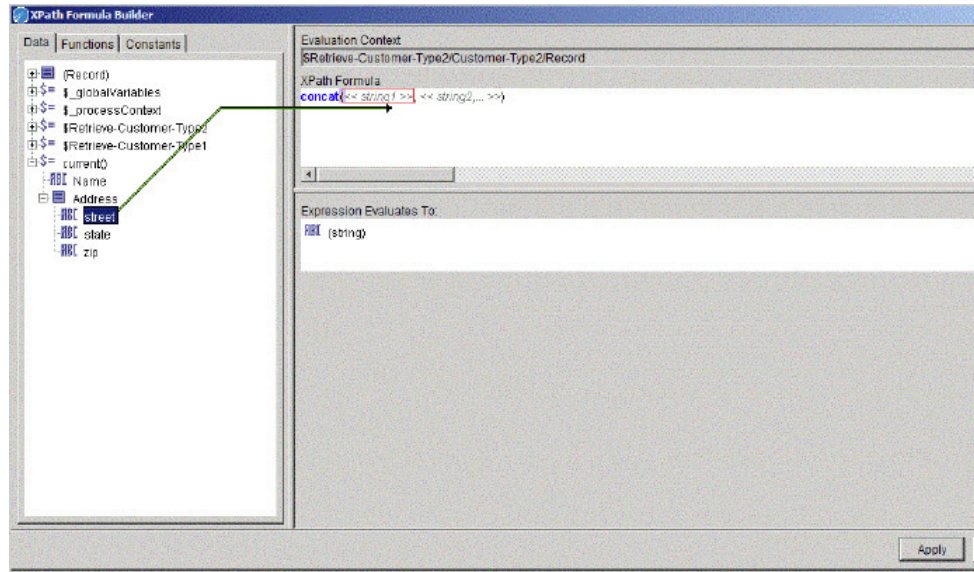
The mapping wizard presents you with an option to automatically map elements with the same name. Click Finish to accept the default mappings.



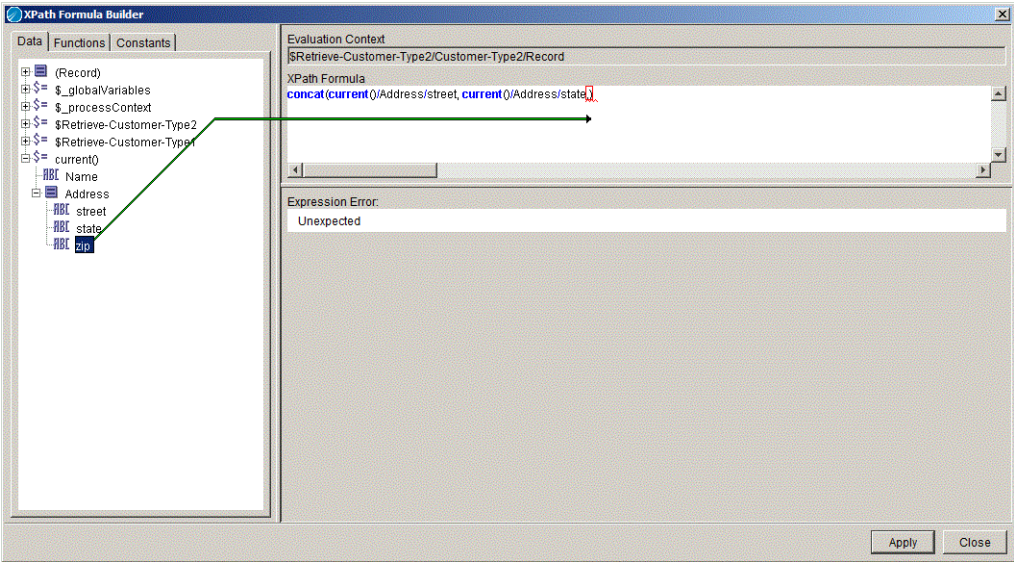
- 4. Select the Address element and click the XPath Formula Builder icon in the Input tab toolbar. In the XPath Formula Builder, drag a concat () function into the XPath Formula field. This function is used to concatenate the three elements in the Record element in the process data area to one Address element in the activity's input.



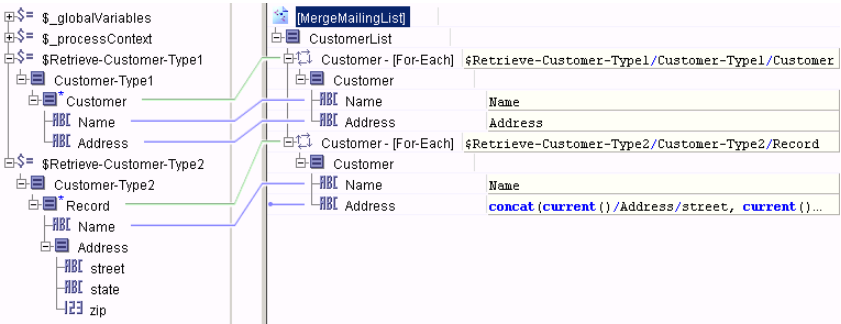
Click the Data tab, then drag the \$current()/Address/street element into the << string1 >> placeholder in the concat() function.



Drag the `$current()/Address/state` element into the `<< string2 >>` placeholder in the `concat()` function. Then, add a comma to the end of the function to include a third string to concatenate. Drag the `$current()/Address/zip` element into the position of the third string in the `concat()` function.



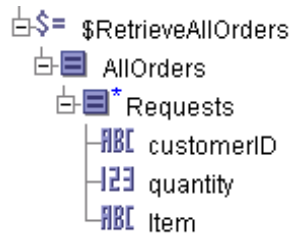
5. Click **Apply**, then click **Close** to dismiss the XPath Formula Builder dialog.
6. This results in the following mapping:



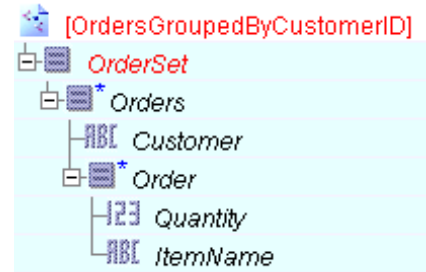
Converting a List Into a Grouped List

You may need to convert a flat list of items into a more structured list. For example, you may have list of all orders that have been completed. You may want to organize that list so that you can group the orders placed by each customer. This scenario typically occurs when you retrieve records from a relational database and the records must be structured differently.

In this example, the schemas are the following:



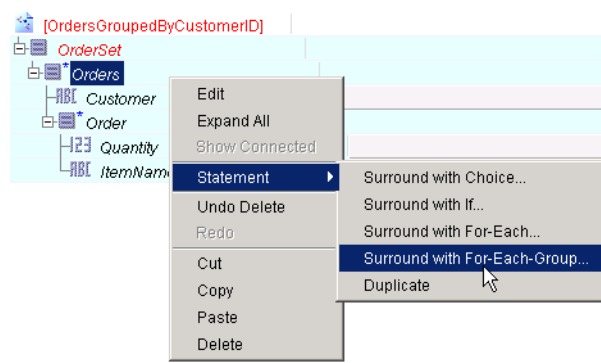
Flat list of orders placed by all customers. Each item in the repeating element Requests lists an order. The same customer can have multiple orders in this list.



The resulting schema is a repeating list of Orders, each item in the list is a customer ID. Each customer ID will, in turn, contain a

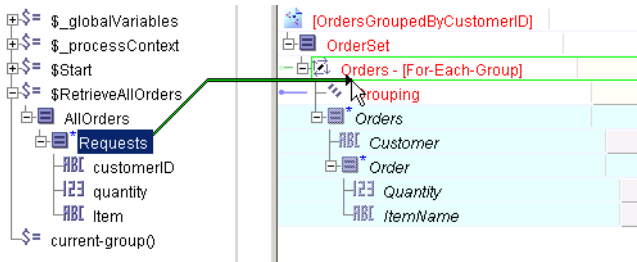
The following procedure describes how to map the flat list of orders into a list grouped by customer ID.

1. Choose the repeating element in the Activity Input schema that holds the grouped data. In this example, that element is **Orders**. Right-click on this element and choose **Statement > Surround with For-Each-Group...** from the pop-up menu. This is a shortcut to create a For-Each-Group statement with the **Orders** element as a child element and a Grouping statement to contain the element you wish to group-by.

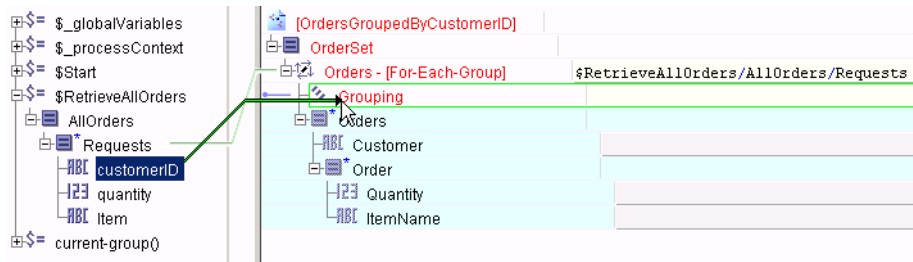


Adding the Grouping statement creates the `$=current-group()` element in the Process Data area. The Grouping statement creates the list grouped by the desired element, and the `current-group()` function allows you to access the items in the **Requests** repeating element that correspond to the group that is currently being processed.

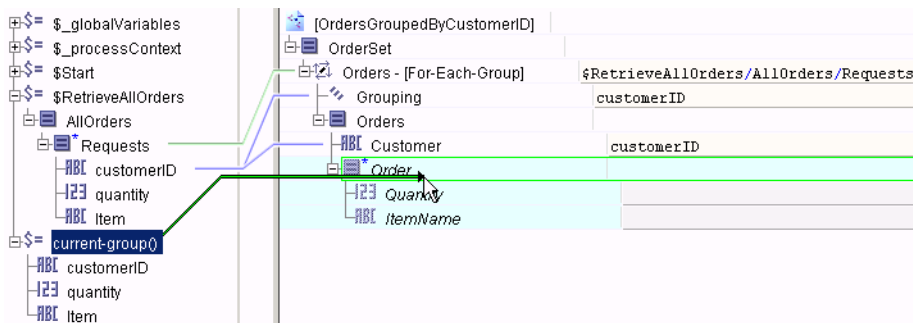
- 2. Drag the repeating element from the Process Data area to the For-Each-Group statement.



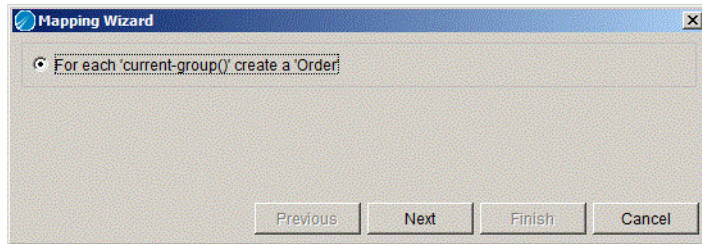
- 3. Drag the element you wish to group by from the Process Data area to the Grouping statement in the Activity Input area. In this example, customerID is the grouping element.



- 4. Map the current-group() element in the Process Data area to the repeating element Order under the Customer element in the Activity Input area.

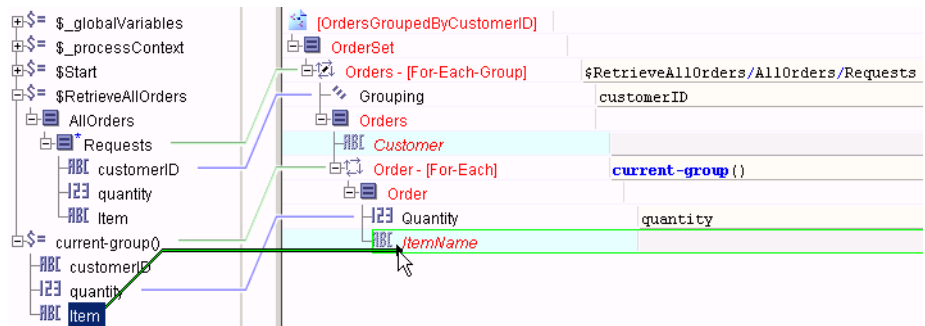


The default choice in the mapping wizard for this mapping is to create a For-Each. Choose this option in the mapping wizard.

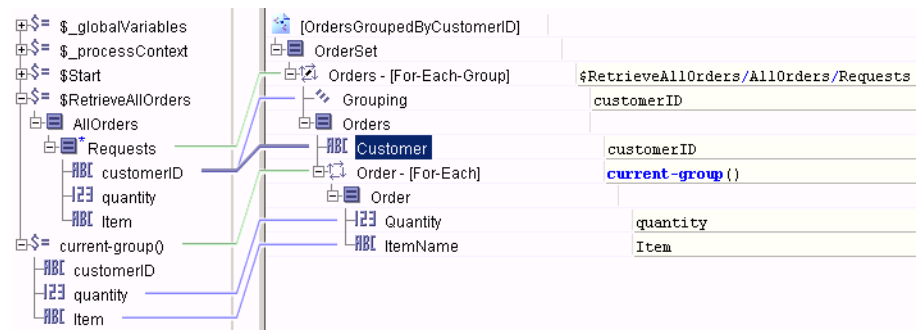


This creates an item in the Order list for each item in the current customer ID group that is being processed. The mapping wizard asks if you wish to map items with the same name in the current group and the orders group.

5. Map the remaining element from the `current-group()` element into the desired element in the For-Each group. In this case, `quantity` would map to `Quantity` automatically, and `Item` must be mapped to `ItemName`.

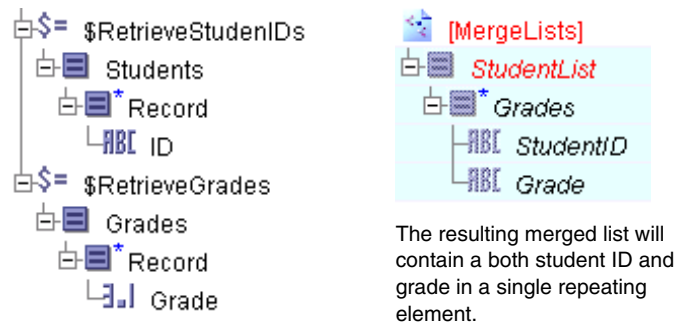


6. Map the `customerID` element in the `Requests` element into the `Customer` element in the Activity Input area.



Merging Two Corresponding Lists

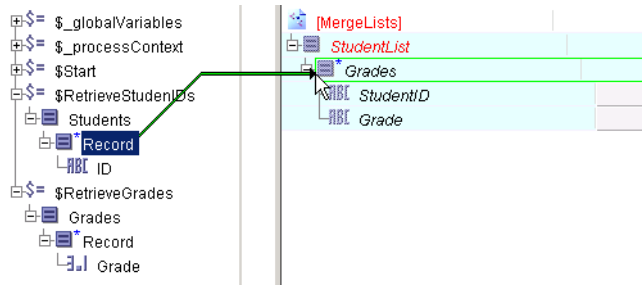
You may need to merge two lists that have corresponding items. For example, you may have a list of student IDs and a list of grades, each grade corresponds to the student ID in the same position in the student ID list. In this example, the schemas are the following:



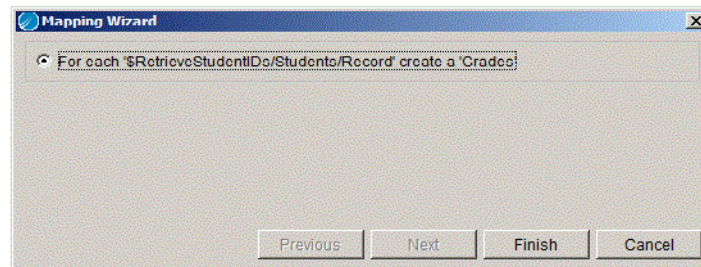
Two repeating lists contain corresponding student IDs and

The following procedure describes how to merge the two repeating elements containing corresponding data into one repeating element.

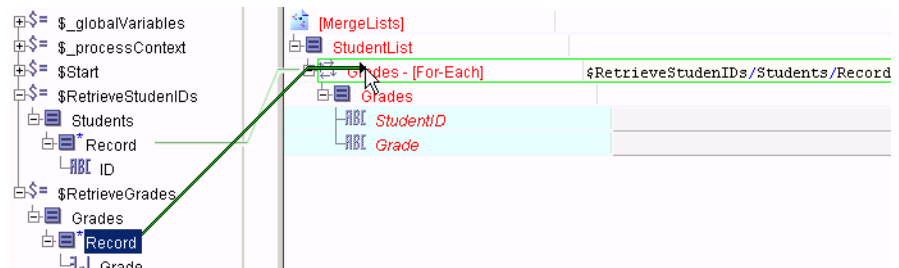
1. Map the first repeating element from the Process Data area into the Grades repeating element in the Activity Input area. In this example, the **\$RetrieveStudentIDs/Students/Record** is the first repeating element.



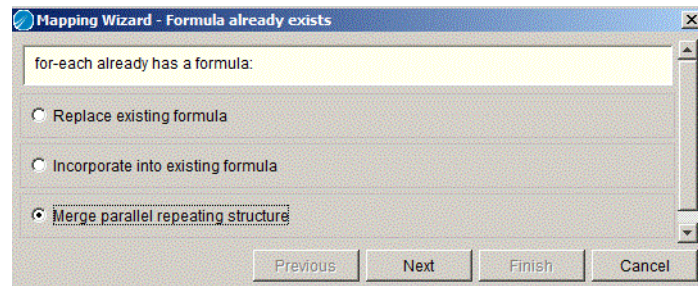
This brings up the mapping wizard with the default choice of creating a For-Each statement. Click **Finish** in the Mapping Wizard dialog to create the For-Each statement.



2. Drag the second repeating element into the For-Each statement.

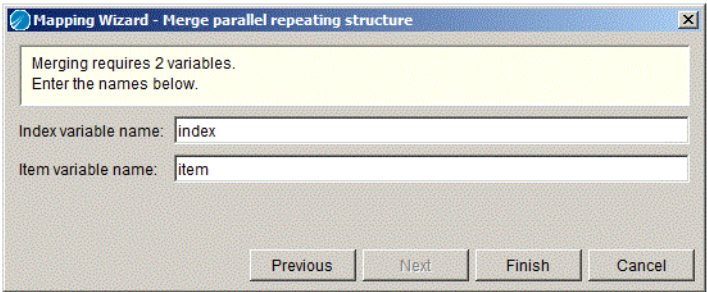


3. The Mapping Wizard dialog appears asking you to choose an option. Choose the Merge parallel repeating structure option and click **Next**.



4. Merging two parallel repeating structures requires two variables. The mapping wizard prompts you to name these two variables. One variable is to hold the position number of the current item being processed, and the other variable is to hold the item in the second list that corresponds to the position of the item in the first list. Create the variables with the default names

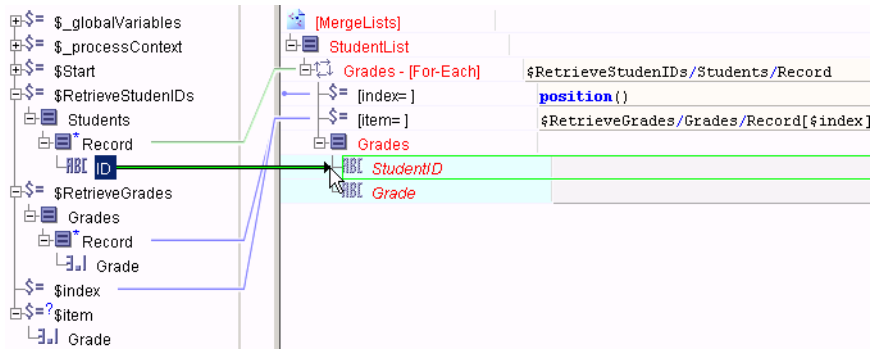
supplied by the mapping wizard, or choose your own names for these variables. Click **Finish** to proceed.



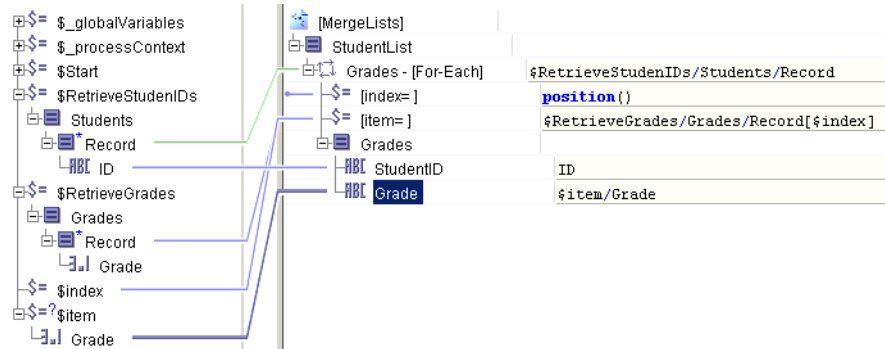
The two variables appear in the Process Data area once you have completed this step. The two variables also appear in the Activity Input area with the correct XPath statement to produce the desired result.

The `$_[index=]` element contains the XPath formula `position()` to set the element with the current position number of the list item being processed. The `$_[item=]` element contains a statement to retrieve the item in the second repeating element that corresponds to the position of the item in the first list that is currently being processed.

5. Map the ID element to the StudentID element in the Activity input.



6. Map the `item/Grade` element to the Grade element in the Activity Input area.

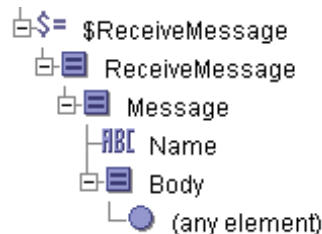


Coercions

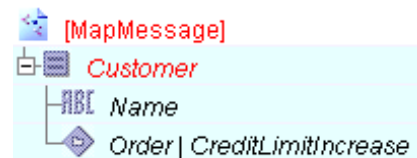
In some situations, the datatype of a Process Data element may be undefined. In these situations, you may know the datatype of the element, and you can coerce the element into a specific type. The Coercions button in the Input tab toolbar allows you to create and manage your coercions.

The following example illustrates a schema with an element defined as the "any element" datatype. The schema is for a generic incoming message that can have any type of body. In the example, however, the any element is coerced into an Order type so that it can be mapped to a choice element.

In this example, the schemas are the following:



The incoming message can have



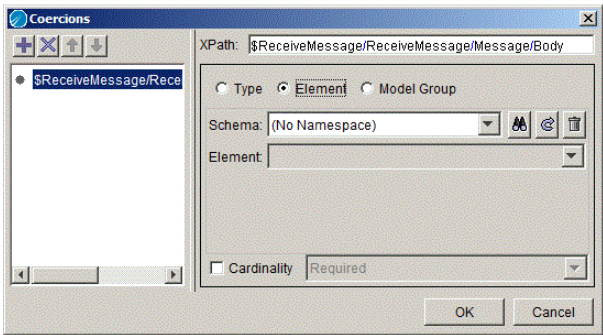
The activity input is expecting either an Order or a CreditLimitIncrease for the

The following procedure describes how to coerce the Body element of the incoming message into a specific datatype and map it to a choice element.



There are many ways of accomplishing the same result as this example. This example attempts to illustrate the simplest method to achieve the desired result.

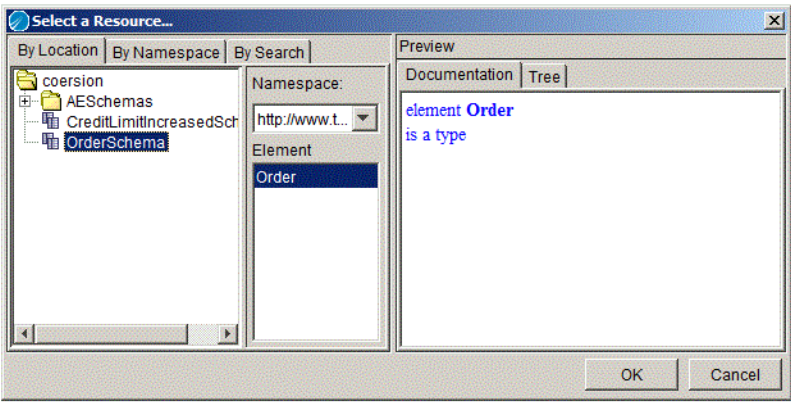
1. Select the element of type any element in the Process Data schema. Click the Coercions button in the Input tab toolbar. In the Coercions dialog, click the Insert button (+) to add a coercion for the currently selected element.



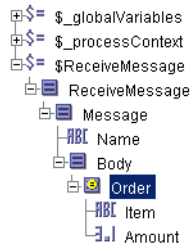
The Coercions dialog allows you to manage all of your coercions for an activity in one dialog. You can create, modify, or delete coercions for any element in the Process Data schema using this dialog, not just the currently selected element. If you are creating a coercion for an element that is not currently selected, use the XPath field to specify the location of the element.

Click the Element radio button to specify that you are specifying a schema element.

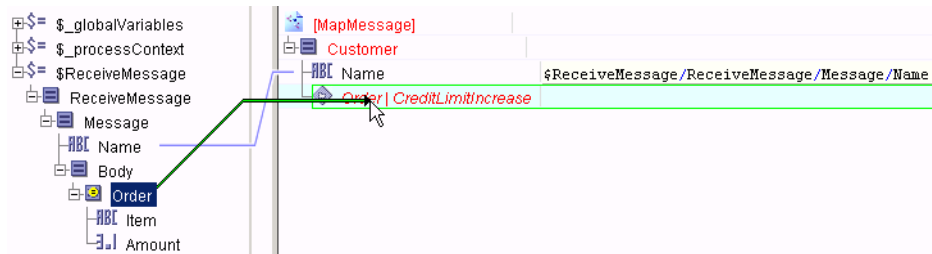
2. Click the Browse Resources button next to the Schema field to browse a list of schemas that can be used. In the Select a Resource... dialog, select the schema that you would like to specify



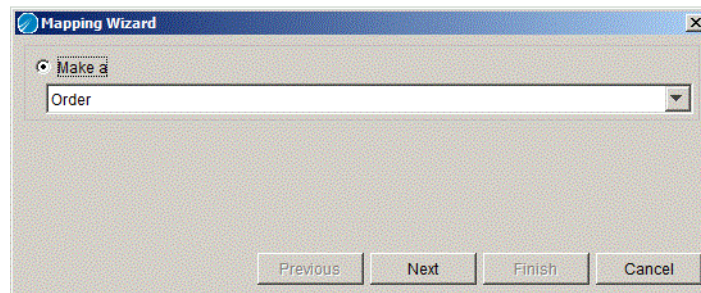
Click OK to coerce the element into the datatype of the selected schema element. The following would be the resulting schema where the element of the datatype any element has been replaced with the `Order` schema.



3. Map the Name element to the Name element in the Activity Input area. Then, map the coerced Order element to the choice element in the Activity Input area.

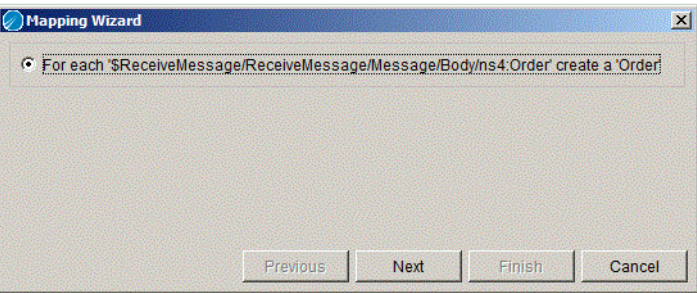


The Mapping Wizard dialog appears and asks if you wish to create an `Order` or a `CreditLimitIncrease` element. Select `Order` and click **Next**.

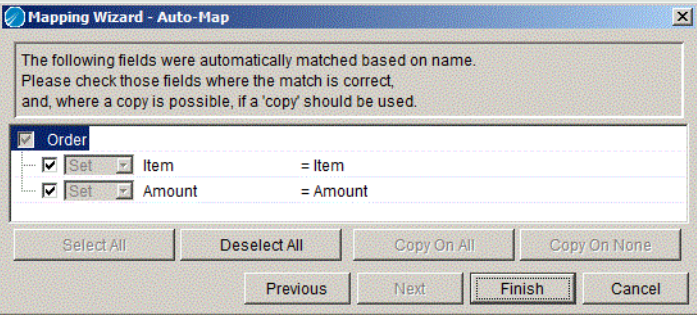


The Mapping Wizard then asks you to create a For Each. Even though there is only one element in the Process Data schema (the `Message` element is not

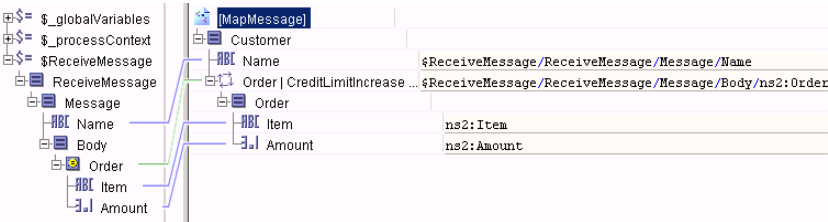
repeating), a For Each is used because this construct allows you to map the individual items of the Order element. Click **Next** to continue.



The Mapping Wizard then asks if you wish to automatically map elements with the same name. Click **Finish** to accept the default mappings.



4. The following is the completed mapping.



XSLT Statements

The following sections describe the XSLT statements you can add to your Activity Input. You can add or edit these statements by clicking the Edit Statement button or these statements can be added automatically by selecting them from the dialogs that appear when you drag and drop elements from the Process Data tree to the Activity Input tree.

Attribute

Allows you to specify an attribute, and optionally the namespace for the attribute. You can also specify the type of value for the attribute.

XSLT Equivalent

The following is an attribute named "lastname".

```
<ns:attribute namespace="mns" name="lastName"/>
```

When attributes are created, you can optionally specify the kind of value the attribute will have and whether the attribute should be surrounded by an if statement. For example, you can specify the value of the last name attribute to be a constant, like so:

```
<ns:attribute namespace="mns" name="lastName"/>
  "Smith"
</ns:attribute>
```

Choose

Provides a way to select transformation to perform based on an expression. Specify the condition in the when element as an XPath expression. You can optionally specify an otherwise condition for processing all elements that do not meet any of the specified when conditions.

XSLT Equivalent

The following determines if the node set for FilesTransferred contains any files, and if so, performs an action. If the node set is empty (no files were transferred), a different action is performed.

```
<ns0:choose xmlns:ns0="http://www.w3.org/1999/XSL/Transform">
  <ns0:when test="$FTP-Put/FTPputOutputFile/FileTransferred" >
    < something here ... >
  </ns0:when>
  <ns0:otherwise>
```

```

        < something here ...>
    </ns0:otherwise>
</ns0:choose>

```

Comment

Places a comment in the XSLT template. Comments are delimited by `<!--` and `-->`.

XSLT Equivalent

```
<!-- comment here -->
```

Copy

Copies the selected node to the current node in the input tree. Only the node is copied, no children of the node are copied.

XSLT Equivalent

```
<ns0:copy xmlns:ns0="http://www.w3.org/1999/XSL/Transform"
select="$Query/resultSet"/>
```

Copy-Contents-Of

Copies the selected node's contents. This is useful if you wish to copy an element to a new element with a different name.

XSLT Equivalent

```
<ns:element namespace="foo" name="bar">
    <ns:copy-of select="null/@*" />
    <ns:copy-of select="null/node()" />
</ns:element>
```

Copy-Of

Creates a copy of the selected node, including the node's children. Both the copied node and the destination node must have the same name and structure.

XSLT Equivalent

```
<ns0:copy-of xmlns:ns0="http://www.w3.org/1999/XSL/Transform"
select="*" />
```


Element

Creates an element with the specified name.

XSLT Equivalent

```
<elementName>value</elementName>
```

For-Each

Performs the specified statements once for each item in the selected node. This is useful if you wish to process each item of a repeating element once.

XSLT Equivalent

The following iterates over the list of files transferred from an FTP Put activity and outputs an element with the name of each file for each file transferred.

```
<ns:for-each select="$FTP-Put/FTPputOutputFile/FileTransferred">
  <fileName>
    <ns:value-of
      select="$FTP-Put/FTPputOutputFile/FileTransferred/Name" />
    </fileName>
  </ns:for-each>
```

For-Each-Group

Groups the items in a list by a specified element. This statement requires a Grouping statement to specify which element to group-by. See [Converting a List Into a Grouped List on page 144](#) for an example of using the For-Each-Group statement.

XSLT Equivalent

```
<ns0:for-each-group
  xmlns:ns0="http://www.w3.org/1999/XSL/Transform" select=""/>
```

Generate Comment

Places a comment element into the XSLT template. This comment will be generated into the activity's output.

Comment elements have the following syntax:

```
<ns0:comment xmlns:ns0="http://www.w3.org/1999/XSL/Transform"/>
```

Generate PI

Places a processing instruction into the XSLT template.

XSLT Equivalent

```
<ns0:processing-instruction
xmlns:ns0="http://www.w3.org/1999/XSL/Transform" name="" />
```

If

An if statement is used to surround other statements in an XSLT template to perform conditional processing. If the test attribute evaluates to true, the statements in the if are output, otherwise they are not output.

XSLT Equivalent

The following if statement surrounds an attribute for processing order items. All items except the last order item are retrieved from the output of the GetOrderInformation activity. The last order item is not output.

```
<ns:if xmlns:ns="http://www.w3.org/1999/XSL/Transform"
test="not(position()=last())">
  <ns:attribute name="OrderItem">
    <ns:value-of select=
      "$GetOrderInformation/OrderInformation/OrderDetails/OrderItem"/>
    </ns:attribute>
  </ns:if>
```

Value-Of

Specifies a value-of statement. This is normally done implicitly by specifying the formula for an activity input item, but you may insert this statement explicitly.

XSLT Equivalent

```
<ns:value-of xmlns:ns="http://www.w3.org/1999/XSL/Transform"
select="" />
```

Variable

Adds a local variable for use in the current mapping. You can specify the name of the variable and whether you wish the variable to have a select attribute.

When you add a local variable, it appears in the Activity Input and Process Data areas. You can supply any XPath expression to the new variable in the Activity Input area (either through mapping or through the XPath Formula Builder).

Once the variable's contents have been supplied, the variable (in the Process Data area) can be mapped to any activity input item.

Adding a variable is useful when you wish to join two repeating elements into a single list, then map the combined list to an activity input item. Adding a variable is also useful if you perform the same computation repeatedly. You can map the results of the computation to several activity input items instead of recreating the computation for each item.

Variables can also improve performance of mappings for large data structures. For example, if you have a process variable with 40 sub-elements, and you map each of the sub-elements to a corresponding input item, TIBCO ActiveMatrix BusinessWorks must retrieve the current process variable for each XPath expression, in this case 40 times. If this mapping appears in a loop, the retrieval of the current process variable occurs 40 times per iteration of the loop. With a variable, the data is retrieved only once and used for all mappings containing the variable. Therefore, to improve performance, create a local variable to hold process variables with a large number of elements and use the local variable in XPath expressions instead of the process variable.

XSLT Equivalent

```
<ns0:variable xmlns:ns0="http://www.w3.org/1999/XSL/Transform"
name="var" select="$RetrieveResults/resultSet"/>
```


Chapter 9 XPath

XPath (XML Path Language) is an expression language developed by the World Wide Web Consortium (W3C) for addressing parts of XML documents. XPath also has basic manipulation functions for strings, numbers, and booleans.

TIBCO ActiveMatrix BusinessWorks uses XPath as the language for defining conditions and transformations. For a complete description of XPath, refer to the XPath specification (which can be obtained from www.w3.org). This section covers the basics of XPath and its use in TIBCO ActiveMatrix BusinessWorks.

Topics

- [XPath Basics, page 162](#)
- [The XPath Formula Builder, page 165](#)
- [String Representations of Datatypes, page 168](#)
- [Date and Time Functions, page 170](#)

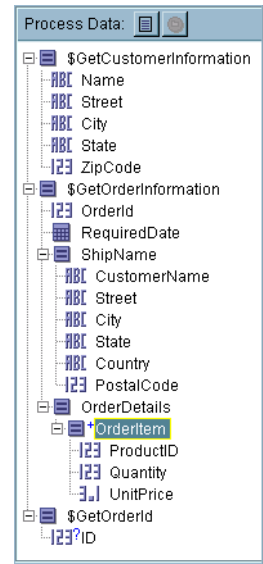
XPath Basics

TIBCO ActiveMatrix BusinessWorks uses XPath (XML Path Language) to specify and process elements of data schema. These data schema are either process variables or input schema for an activity. You can also use XPath to perform basic manipulation and comparison of strings, numbers, and booleans. To use XPath in TIBCO ActiveMatrix BusinessWorks, you need only be familiar with the basic XPath concepts, but you may wish to learn more about XPath when building complex expressions. For a complete description of XPath, refer to the XPath specification (which can be obtained from www.w3.org).

Addressing Schema Elements

All process data and activity input are represented as an XML schema. Regardless of where the data comes from or its format, TIBCO ActiveMatrix BusinessWorks represents the data as a schema tree. The data can be simple (strings, numbers, booleans, and so on), or it can be a complex element. Complex elements are structures that contain other schema elements, either simple elements or other complex elements. Both simple and complex elements can also repeat. That is, they can be lists that store more than one element of the given type.

XPath is used to specify which schema element you would like to refer to. For example, the following schema may be available for an activity's input:



The process data area of the example input tab illustrates the output schema of the activities in the process. There are three output schema, each a root node in the process data area: `GetCustomerInformation`, `GetOrderInformation`, and `GetOrderId`. Each of these schema has its own associated structure, for example, `GetCustomerInformation` has a set of simple values and `GetOrderInformation` has simple data and other complex data.

To reference a particular data item in any of these schema, you start with the root node and then use slashes (/) to indicate a path to the desired data element. For example, if you wish to specify the `Street` attribute in the `ShipName` complex element that is in the `GetOrderInformation` node, you would use the following syntax:

```
$GetOrderInformation/ShipName/Street
```

The path starts with a dollar sign to indicate it begins with a root node, then continues with node names using slashes, like a file or directory structure, until the desired location is named.

Evaluation Context

XPath also has a method for referencing relative paths from a particular node. If you have an *evaluation context*, or a particular starting node in a schema tree, you can specify the relative path to other elements in the tree.

For example, if your evaluation context is `$GetOrderInformation/ShipName`, then you can reference the sub-items of `ShipName` without specifying the entire path. If you wish to reference `$GetOrderInformation/RequiredDate`, the relative path would be `../RequiredDate`. The path is relative to the evaluation context — `RequiredDate` is one level higher in the schema tree than the elements of `ShipName`.

Namespaces

Some schema elements must be prefixed with their namespace. The namespace is automatically added to elements that require this when creating mappings on the Input tab of an activity or when dragging and dropping data in the XPath formula builder.

Search Predicates

An XPath expression can have a search predicate. The search predicate is used to locate a specific element of a repeating schema item. For example, the `$GetOrderInformation/OrderDetails/OrderItem` item is a repeating element. If you wish to select only the first item in the repeating element, you would specify the following:

```
$GetOrderInformation/OrderDetails/OrderItem[1]
```

The `[1]` specifies the first element of a repeating item.

Sub-items can also be examined and used in a search predicate. For example, to select the element whose `ProductId` is equal to "3A54", you would specify the following:

```
$GetOrderInformation/OrderDetails/OrderItem[ProductId="3A54"]
```

You can also use functions and expressions in the search predicate. For example, if you wish to find all elements after the first, you would specify the following:

```
$GetOrderInformation/OrderDetails/OrderItem[position()>1]
```

See the online documentation available in the XPath formula builder for a list of the available operators and functions in XPath.

You can also build custom Java functions and make them available in XPath by using the Java Custom Function shared resource. See the description of the Java Custom Function shared configuration resource in *TIBCO ActiveMatrix BusinessWorks Palette Reference* for more information about creating custom functions and making them available in XPath.

Testing For Nil

Some elements can be explicitly set to nil. You can test an element to determine if it is set to nil or not. For example, the following XPath expression returns true if the `$Order/Item/OnSale` element is set to nil:

```
$Order/Item/OnSale/@xsi:nil="true"
```

Comments

You can add comments to XPath expressions using the XPath 2.0 syntax for comments. The syntax is:

```
{-- <comment here> --}
```

For example, the following XPath expression contains a comment:

```
$GetOrderInformation/ShipName/Street {-- returns the street --}
```


The XPath Formula Builder

The XPath formula builder can be used where XPath expressions are allowed, such as when creating transformations on the Input tab of an activity. The XPath formula builder allows you to drag and drop schema elements and XPath functions to create XPath expressions. The schema elements, when dragged into the XPath Formula field, automatically become valid XPath location paths for the desired item. If a function is dragged into the XPath formula window, there are placeholders for each parameter of the function. You can drag and drop schema elements over the parameter placeholders to replace each placeholder.

Figure 29 illustrates using the XPath formula builder to drag and drop schema elements into function placeholders.

Figure 29 The XPath formula builder

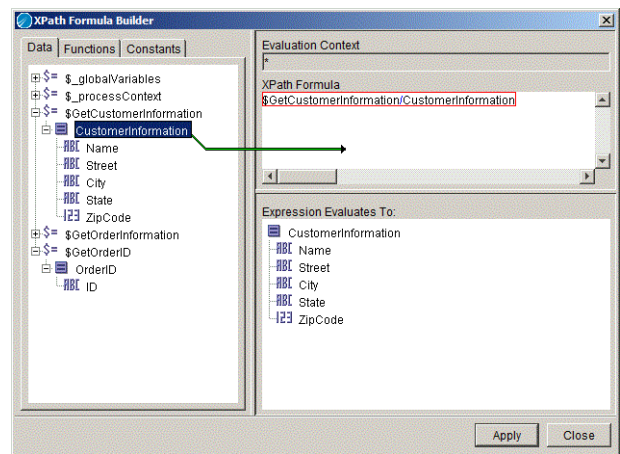


Table 15 describes the different areas of the XPath formula builder.

Table 15 XPath formula builder elements

Element	Description
Data tab	Displays the process data schema tree. All elements in this tree are available to drag and drop into the XPath Formula field.

Table 15 XPath formula builder elements

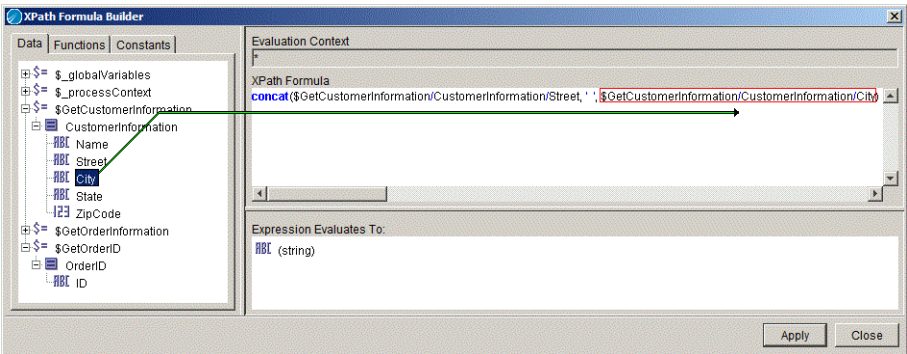
Element	Description
Functions tab	Displays the available XPath functions. These are categorized into groups and each function can be dragged from the function list into the XPath Formula field. When the function is placed into the XPath formula, placeholders are displayed for the function's parameters. You can drag and drop schema elements from the Data tab into the function's placeholders. The result of evaluating the function is displayed in the "Expression Evaluates To" panel. If there are any errors in the expression, they are listed there as well. For more information about XPath functions, see the description of the function that is displayed when it is selected in the XPath formula builder.
Constants tab	Displays the constants available for use in XPath expressions. These are categorized into groups and each constant can be dragged from the constants list into the XPath Formula field. Constants are useful for inserting special characters, such as tabs, symbols, and so on, into XPath formulas. Constants are also defined for commonly used items, such as date formats.
Documentation panel	Describes each selected function. As you click on a function in the Function tab, the documentation panel gives a brief description of the function and one or more examples.
Evaluation Context field	Displays the evaluation context of the expression field that the editor was invoked from. See Evaluation Context on page 163 for more information about the evaluation context.
XPath Formula field	Displays the XPath formula you wish to create. You can drag and drop items from the Data tab or the Functions tab to create the formula.

Table 15 XPath formula builder elements

Element	Description
Expression Evaluates To Panel	Displays the result of evaluating the formula shown in the XPath Formula field. If there are errors in the formula, they are displayed here.

Figure 30 illustrates using the XPath formula builder to create a valid function. The function concatenates the data elements `$GetCustomerInformation/Street` and `$GetCustomerInformation/City` and places a space between the two elements.

Figure 30 Creating an XPath formula



String Representations of Datatypes

When data must be represented in the input or output of an activity, the data is represented as a string. For example, writing the output of an activity to a file requires that numeric data be serialized into string form. Also, supplying a floating point number to an XPath function in the Input tab of an activity often involves typing the number as a string into the input element. TIBCO ActiveMatrix BusinessWorks follows the XPath 1.0 standard for representing all numeric datatypes. TIBCO ActiveMatrix BusinessWorks follows the XML Schema canonical format for all other datatypes.

The following sections describe the string representations of datatypes.

Numeric Datatypes	Numeric datatypes include all types derived from <code>xs:integer</code>, <code>xs:decimal</code>, <code>xs:float</code>, and <code>xs:double</code>. All decimal, float, and double numbers are compressed to an integer when represented, if there are only zeros following the decimal point (for example, "1.000" is represented as 1). Scientific notation is never used to represent a floating point number as a string (for example, "<code>xs:double('1.234E05')</code>" is represented as 123400). Data is truncated if the number of digits exceeds the maximum precision for the datatype (for example, "<code>xs:float('1.23456789')</code>" is represented as 1.2345679). Both zero and negative zero are represented as 0. Positive and negative infinity are represented as <code>Infinity</code> and <code>-Infinity</code>. Not a number is represented as <code>NaN</code>.
Boolean	The boolean datatype is used to indicate a true or false state. <code>xs:boolean('true')</code> and <code>xs:boolean('1')</code> are represented by <code>true</code>. The XPath function <code>true()</code> is also represented as <code>true</code>. <code>xs:boolean('false')</code> and <code>xs:boolean('0')</code> are represented by <code>false</code>. The XPath function <code>false()</code> is also represented as <code>false</code>.
Date Datatypes	Activities in TIBCO ActiveMatrix BusinessWorks implement dates in one of two ways. Either a date is stored as the number of milliseconds since January 1, 1970, or the date is implemented according to the XPath 2.0 or XQuery 1.0 standards as a set of normalized components (<code>xs:date</code>, <code>xs:time</code>, <code>xs:dateTime</code>, and so on) with an optional time zone offset. Activities that are associated with Java (for example, Java Code, Java Method, and so on) use the first implementation. Activities that are associated with XML (for example, Mapper, Parse XML, and so on) use the second implementation. The second implementation supports arbitrary precision of the seconds component.

Conversion between these representations may result in a loss of information either because of the difference in time zone representation or the precision of the seconds.



Dates output by the Java Code activity contain the time zone. Other Java activities (Java Method, Java to XML, and so on) do not use the time zone. Therefore mapping a date output by a Java Code activity to the input of any other Java activity will result in the loss of the time zone information.

Date and Time Functions

There are some functions in the XPath formula builder that allow you to parse or format strings that represent dates and times. These functions are:

- `format-dateTime(<<format>>, <<dateTime>>)`
- `format-date(<<format>>, <<date>>)`
- `format-time(<<format>>, <<time>>)`
- `parse-dateTime(<<format>>, <<string>>)`
- `parse-date(<<format>>, <<string>>)`
- `parse-time(<<format>>, <<string>>)`

The *format* parameter of these functions is based on the format patterns available for the `java.text.SimpleDateFormat` Java class. In the format parameter, unquoted alphabetic characters from A to Z and a to z represent the components of the date or time string. You can include non-pattern alphabetic characters in the string by quoting the text with single quotes. To include a single quote, use `'`. [Table 16](#) describes the alphabetic characters and their associated presentation in a date or time string.

Table 16 *Formatting characters in date or time strings*

Letter	Description	Example
G	Era Four or more Gs return the full name of the era.	AD
Y	year Two ys return two-digit year.	2003; 03
M	Month in year Three or more Ms return text name.	August; Aug; 08
w	Week in year	48
W	Week in month	3
D	Day in year	254
d	Day in month	28

Table 16 Formatting characters in date or time strings

Letter	Description	Example
F	Day of week in month	3
E	Day in week Four or more Es return the full name of the weekday.	Friday; Fri
a	AM/PM marker Four or more as return the full name.	AM
H	Hour in day (0-23)	23
k	Hour in day (1-24)	1
K	Hour in AM/PM (0-11)	11
h	Hour in AM/PM (1-12)	1
m	Minute in hour	59
s	Second in minute	48
S	Millisecond	456
z	Time zone represented as a GMT offset.	GMT-08:00
Z	RFC 822 four-digit time zone format	-0800
all other letters	Reserved	—

For any format pattern letter that returns a numeric value (for example, w, h, and m), the number of letters in the format pattern represents the minimum number of digits. For formatting functions, if the date or time has fewer digits than the number of pattern letters, the output is padded with zeros. For parsing functions, when the date or time has fewer digits than the number of characters in the format pattern, the extra characters are ignored, unless they are needed to determine the boundaries of adjacent fields.

Table 17 illustrates some example date and time format patterns and the resulting string.

Table 17 Example date and time format patterns

Date/Time Pattern	Result
"yyy.MM.dd G 'at' HH:mm:ss"	2003.3.11 AD at 09:43:56
"EEE, MMM d, ''yy"	Tue, Mar 11, '03
"hh 'o''clock' a, zzzz"	9 o'clock AM, GMT-8:00
"K:mm a"	0:08 PM
"yyMMddHHmmssZ"	010704120856-700

BW Process Information Functions

BusinessWorks provides functions in the XPath formula builder that can be used to fetch process related information for any activity. These functions, listed under the group BW, are:

- `getCurrentProcessName(<processID>)`: Fetches the process name associated with the specified `<processID>`
- `getCurrentActivityName(<processID>)`: Fetches the activity name associated with the specified `<processID>`
- `getHostName()`: Fetches the host name on which the process is running. This function does not take any input parameter.
- `getFileStream()`: Reads binary data directly from a file in base64 field if MTOM is used.

The `<processID>` parameter has to be specified when you choose one of the process information functions.

Chapter 10 **Error Handling**

When executing business processes, activities can encounter errors. You may wish to add procedures to your process definitions for handling any expected or unexpected errors. This chapter describes error handling in process definitions.

Topics

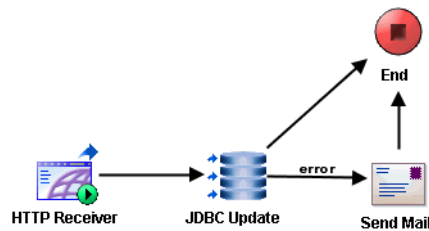
- [Overview of Error Handling, page 175](#)
- [The Error Process Variables, page 176](#)
- [Error Propagation, page 179](#)
- [Process Error Schemas, page 181](#)
- [Using the Catch and Rethrow Activities, page 183](#)

Overview of Error Handling

Errors can occur during activity processing. For example, an error may occur during a Send Mail activity if the specified mail host does not exist. You can specify that one transition out of an activity is to be taken in the case of an error. You then specify activities you wish to execute in the event of an error. This allows you to create error-handling procedures for dealing with potential runtime errors in your process definitions.

For example, the following illustrates a simple process that begins with an HTTP request and updates a database based on the incoming request. If the update is successful, the process ends. If an error is encountered (for example, the database is down), an email is sent to a system administrator, and then the process ends. [Figure 31](#) illustrates this simple error-handling procedure. The error transition is used to specify the activities to execute in case of an error.

Figure 31 A simple error-handling procedure



Error handling can also involve significantly more complex processing. The following sections describe error handling in more detail.

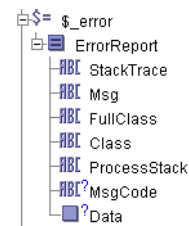
The Error Process Variables

When an error transition is taken, error data is available to activities that execute after the error transition. There are two types of process variables that contain error data, the `$_error` process variable and the activity error variables. Activity error variables are named `$_error_<activityName>`, and all activities that have an error transition are available in the Process Data panel after an error transition is taken. You can use the data in these process variables to perform the desired processing, depending upon what error occurred.

The following sections describe the two kinds of error process variables.

`$_error` Process Variable

The `$_error` process variable contains general information about the process in which the error occurred. The schema of the `$_error` variable is the following:

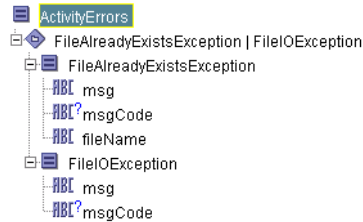


The contents of each schema item are dependent upon the activity that throws the error. The Data schema item contains activity-specific error information. You can use a coercion to change the Data element into the specific type for the activity.

When you create an error-handling procedure, you may find the data in the `$_error` process variable useful. You can map data from this process variable into Input items for activities in your error-handling procedure.

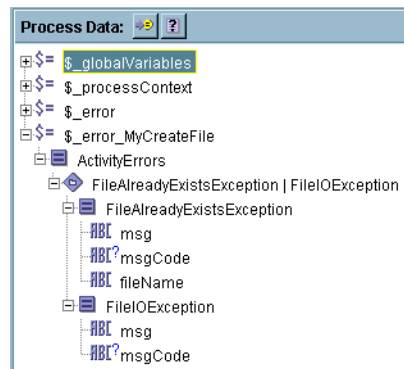
`$_error_<activityName>` Process Variables

Each activity has one or more predefined error schemas. The error schemas are displayed on the activity's Error Output tab. Typically, there are two or three types of exceptions an activity can encounter, and these are represented as a choice element in the error output schema. For example, the following illustrates the error output schema of the Create File activity:



This activity can encounter the `FileAlreadyExistsException` or the `FileIOException`. *TIBCO ActiveMatrix BusinessWorks Palette Reference* describes the error schemas on the Error Output tab of all activities and lists possible causes for each error type. Refer to the description of each activity for more information about the Error Output tab.

When an error is encountered, the `$_error_<activityName>` process variable is created and populated with the error data. The name of the process variable is dependent upon the activity's specified name. You can obtain the error by examining the `$_error_<activityName>` process variable in subsequent activities after an error transition. The following illustrates the Process Data panel that is available after the activity named `MyCreateFile` encounters an error:



You can use XPath expressions to determine which error occurred and handle the error accordingly. For example, in the above error schema, if the `FileAlreadyExistsException` error occurs, then that node is not empty. The following expression can be used to determine if the `FileAlreadyExistsException` occurred:

```
boolean($_error_MyCreateFile/ActivityErrors/  
ns2:FileAlreadyExistsException)
```

You can use the `msgCode` element to determine the kind of error that occurred. *TIBCO ActiveMatrix BusinessWorks Error Codes* lists all error codes that activities can return. You should use the error code instead of the error message text to detect and handle errors. The error code should remain constant from release to release, but the text of the message may change.

Error Propagation

Groups and called processes define the scope of an exception. An exception that occurs within a group or a called process causes processing to halt. Any unhandled exceptions are propagated to the next highest exception scope. Unhandled errors occur where there is no error transition or Catch activity that specifies the activities to execute in the case of an error.

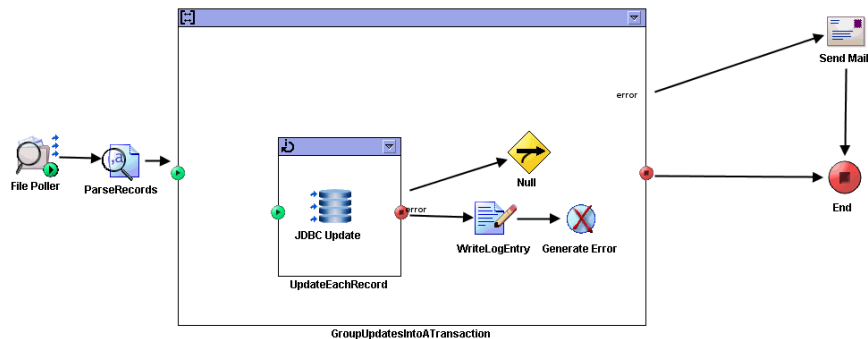
Also, you can use the Generate Error activity to create an unhandled error. The Generate Error activity does not permit any transitions to another activity, so any error created by the Generate Error activity is propagated to the next highest exception scope.

The following sections describe propagation of errors for groups and called processes.

Group Error Propagation

Unhandled errors halt the execution of a group and the error transition out of the group is taken. [Figure 32](#) illustrates a process definition that waits for new text files, parses the files into an XML schema, then inserts the records into a database table.

Figure 32 Propagating errors from a group



The process definition uses two group activities. The first group is an iterate group that performs one update for each record. If any of the updates fail, an error transition out of the group is taken to the WriteLogEntry activity. A second group surrounds the iterate group to enclose all updates in a transaction. If the transaction succeeds, the process ends. If the transaction fails, the error transition is taken out of the transaction group to the SendMail activity.

The Generate Error activity is used to propagate an error outside of the transaction group to the next exception scope. If the iterate group experiences an error, the WriteLogEntry activity is executed, then the error transition out of the group is taken to the Send Mail activity.

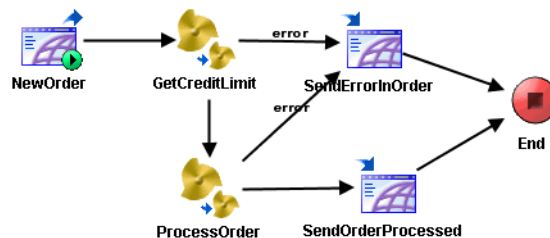
The transition to the Send Mail activity is taken if there is either an error when committing the transaction or if the Generate Error activity is executed (because of an error in the iterate group). The error process variables contain the error information for the activity where the error occurred.

The Generate Error activity can use any error schemas defined on the process to propagate a specific schema to the parent process. See [Process Error Schemas on page 181](#) for more information about process error schemas.

Called Process Error Propagation

When a process definition calls another process definition, the called process can encounter errors. Any unhandled errors encountered when executing the called process cause the called process to halt execution and return the error to the next highest exception scope. [Figure 33](#) illustrates a process definition that waits for an incoming HTTP request that contains an order.

Figure 33 Propagating errors from a called process



The GetCreditLimit process is called to check the credit limit of the customer that places the order. If the credit limit check succeeds, the ProcessOrder process is called. If the order processing is successful, a response is sent back to the customer stating the order is complete and the process definition terminates.

If the GetCreditLimit or ProcessOrder processes encounter an error, a response is sent back to the customer stating there was an error in the order and the process definition terminates.

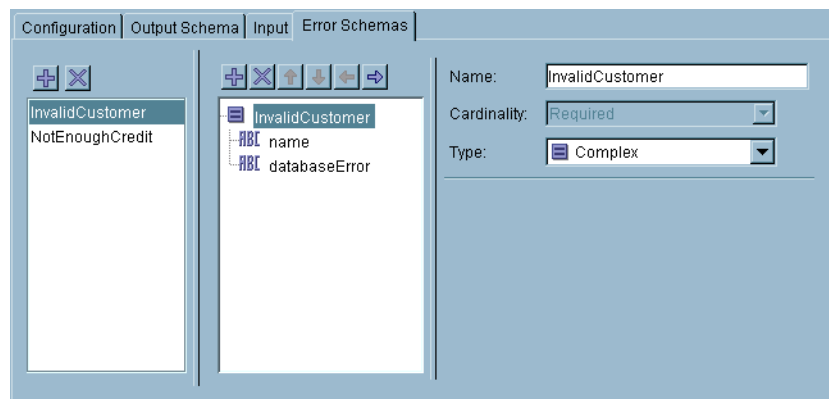
The error process variables contain the error information for the activity where the error occurred. Also, a process can define an error schema and use the Generate Error activity to propagate specific data to the parent process. See [Process Error Schemas on page 181](#) for more information about process error schemas.

Process Error Schemas

The error process variables contain the default data returned in the event of an error. You can define specific error schemas to hold error data when errors are propagated from a group or a called process. Each process can define a number of error schemas by creating these schemas on the Error Schema tab of the process definition's [End Activity](#).

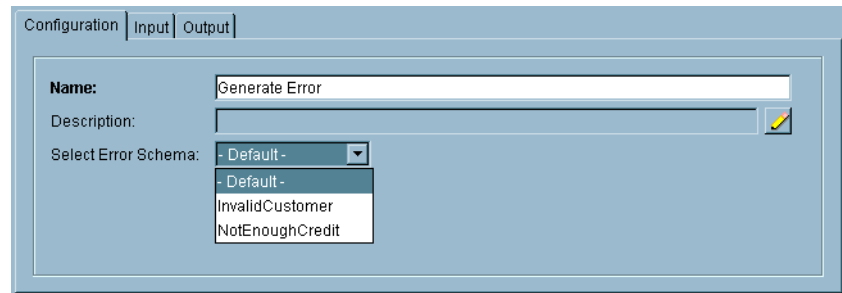
Error schemas are created like any other schema (see [Editor on page 56](#)). However, the Error Schema tab of the End activity allows you to create more than one error schema. [Figure 34](#) illustrates the Error Schemas tab with two error schemas. The left panel of the tab allows you to create or delete schemas. The middle portion allows you to modify the selected schema. The right panel of the tab allows you to modify each schema item.

Figure 34 The Error Schemas tab



Error schemas are used by the Generate Error activity. When the Generate Error activity is executed, the specified error schema is propagated to the parent process. [Figure 35](#) illustrates the Configuration tab of the Generate Error activity. The Select Error Schema field contains a drop-down list of error schemas defined for the process.

Figure 35 The Generate Error Configuration tab



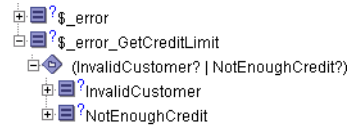
If - Default - is chosen for the error schema, only the `$_error` process variable contains the propagated error data.

If a process error schema is chosen, the schema appears in the Input tab for the Generate Error activity and you can map data to the specified error schema. At run time, the Generate Error propagates the specified error schema to the parent process in the `$_error_<activity-name>` process variable.

The process variable is derived from the activity where the Generate Error occurred. If the Generate Error occurs in a called process, the `<activity-name>` portion of the process variable is the name of the Call Process activity. If the Generate Error occurred in a group, the `<activity-name>` portion is the name of the Generate Error activity in the group.

In the example described in [Called Process Error Propagation on page 180](#), the `SendErrorInOrder` activity has access to the error schema supplied by any `GenerateError` activity in the `GetCreditLimitProcess`. This process specifies two error schemas, `InvalidCustomer` and `NotEnoughCredit`. [Figure 36](#) illustrates the process data available to the `SendErrorInOrder` activity.

Figure 36 Example of process data for error schemas



The available error schemas for the `GetCreditLimit` process are presented as a schema item of type Choice. This item will contain either the `InvalidCustomer` or the `NotEnoughCredit` error schema. You can use XPath to determine which schema is actually contained in the element, and then map the data in the schema accordingly.

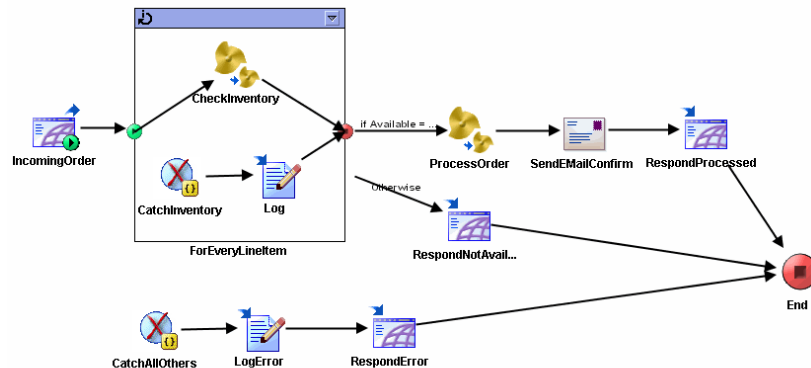
See *TIBCO ActiveMatrix BusinessWorks Palette Reference* for more information about the Generate Error activity.

Using the Catch and Rethrow Activities

You can place a Catch activity in your process definition to deal with unhandled exceptions. The Catch activity allows you to create a track that handles the exception and proceeds to the end of the current scope; either the end of the process definition or the end of a group. You can use the Catch activity as an alternative to individually handling exceptions for each activity, or you can use error transitions to handle some exceptions and the Catch activity to handle others.

Figure 37 illustrates the Catch activity. The process waits for incoming orders sent by way of HTTP requests. When an order arrives, each line item is checked for availability in the `ForEveryLineItem` group. If an error occurs while checking the inventory, execution transfers to the `CatchInventory` activity. A log file entry is written, and then the transition is taken to the end of the group. If the inventory is available, the order is processed, a confirmation email is sent, and the response is sent back to the HTTP client. If the inventory is not available, a response is sent back to the HTTP client stating that one or more items are not available. If an error occurs outside of the `ForEveryLineItem` group, execution transfers to the `CatchAllOthers` activity.

Figure 37 Example of using the Catch activity



The Catch activity can specify the type of exception that should be caught. A list of exceptions that can be raised in the current scope are available on the Configuration tab of the Catch activity. Rethrow activity rethrows an exception but to catch it your second catch activity, Rethrow activity should be in a group. Any exceptions that are not already handled by an error transition or Catch activity can be handled by a Catch activity that specifies the Catch All option on the Configuration tab.

Transitions cannot be drawn to a Catch activity. Also, you cannot draw transitions from any activity on the path of the Catch activity to any activity on the regular execution path. Activities on a Catch path can, however, have transitions to other activities on other Catch paths.

The Rethrow activity allows you to throw the exception currently being handled by a Catch path. This is useful if you wish to perform some error processing, but then propagate the error up to the next level.

See *TIBCO ActiveMatrix BusinessWorks Palette Reference* for more information about the Catch and Rethrow activities.

Chapter 11 Transactions

TIBCO ActiveMatrix BusinessWorks allows you to group some activities into a transaction group. Transactions ensure that all participants in the transaction either complete or are rolled back together. This chapter describes how to incorporate transactions into your process definitions.

Topics

- [Overview of Transactions, page 186](#)
- [JDBC, page 188](#)
- [Java Transaction API \(JTA\) UserTransaction, page 189](#)
- [XATransaction, page 191](#)
- [JMS Local Transaction, page 199](#)
- [Summary of Transactions, page 204](#)

Overview of Transactions

In TIBCO ActiveMatrix BusinessWorks a transaction is a logical unit of work. Transactions allow you to group multiple activities into an atomic execution unit. All activities that can participate in a transaction must either complete successfully or be undone or rolled back together. To create a transaction, you use a group to surround the activities in the transaction.

Not all TIBCO ActiveMatrix BusinessWorks activities can participate in a transaction. Only the following types of activities have transactional capabilities:

- JDBC activities
- JMS activities
- ActiveEnterprise Adapter activities that use JMS transports
- EJB activities
- TIBCO iProcess BusinessWorks Connector activities

Although only the activities above can be part of a transaction, any activity can be contained in a transaction group. For example, you may have three JDBC Update activities and one Write File activity in a transaction group. All the JDBC Update activities either complete or roll back at the end of the transaction. The Write File activity, however, does not participate in the transaction and executes whether the transaction commits or fails.

For more information about creating groups, see [Grouping Activities on page 77](#).

Types of Transactions

TIBCO ActiveMatrix BusinessWorks offers a variety of types of transactions that can be used in different situations. You can use the type of transaction that suits the needs of your integration project. When you create a transaction group, you must specify the type of transaction. TIBCO ActiveMatrix BusinessWorks supports the following types of transactions:

- [JDBC](#)
- [Java Transaction API \(JTA\) UserTransaction](#)
- XA Transaction
- JMS Local Transaction

A transaction group can have only one transaction type. The activities that can participate in the transaction are determined by the transaction type, the configuration of the activity, and the type of the activity. For example, only JDBC activities whose JDBC Connection type is "JDBC" can participate in a JDBC transaction. The following sections describe each of these types of transactions and the situations in which they are used in more detail.

JDBC

The JDBC transaction allows multiple JDBC activities that access the same database connection to participate in a transaction. Only JDBC activities that use the same JDBC Connection participate in this transaction type, but other activities can be part of the transaction group. If the transaction commits, all JDBC activities using the same JDBC connection in the transaction group commit. If the transaction rolls back, all JDBC activities using the same JDBC connection in the transaction group roll back.

The transaction group commits automatically if all activities in the group complete and a non-error transition is taken out of the transaction group. If any errors occur while processing the activities in the group, even errors in non-JDBC activities, the transaction is rolled back and an error is returned (you should have an error transition out of the group to handle this situation).

Individual JDBC activities can override the default transaction behavior and commit separately. See the description of the JDBC palette in *TIBCO ActiveMatrix BusinessWorks Palette Reference* for more information about using JDBC activities.

Multiple JDBC Connections In Transaction Groups

All activities that use the same JDBC Connection shared configuration resource are part of the same transaction. It is possible to use more than one JDBC Connection in the same transaction group. However, only activities that use the same JDBC Connection are guaranteed to commit or rollback together when the transaction completes.

If you have more than one JDBC Connection in the transaction group, each set of activities that uses a JDBC Connection is considered a separate transaction. For example, you have three JDBC Updates in a transaction group, A, B, and C. A and B use JDBC Connection X, but C uses JDBC Connection Y. In this case, the updates for activities A and B are part of one transaction and the update for activity C is part of a different transaction.

To create a distributed transaction across multiple databases, use the XA transaction type.

Configuring JDBC Transactions

To configure a JDBC transaction, select JDBC Transaction as the transaction type of the group. Also, in the JDBC Connection resource(s) used by JDBC and Checkpoint activities in the group, select JDBC in the Connection Type field.

Java Transaction API (JTA) UserTransaction

The Java Transaction API (JTA) UserTransaction type allows JDBC, JMS, ActiveEnterprise Adapter (using JMS transports), and EJB activities to participate in transactions. JTA specifies standard Java interfaces between a transaction manager and the parties involved in a distributed transaction system: the resource manager, the application server, and the application. Sun Microsystems developed and maintains the API. See <http://java.sun.com/products/jta/> for more information about the API.



For activities that use the JMS transport, request/reply operations cannot participate in a JTA transaction.

Using this type of transaction requires an installed and configured application server that implements the JTA interface `javax.transaction.UserTransaction`. For example, the following transaction servers implement the JTA transaction interface:

- BEA WebLogic Application Server
- IBM Websphere Application Server
- JBoss Application Server



Not all application servers permit JMS and JDBC operations to participate in the JTA transaction. Refer to your application server documentation for more information about supported operations. If the application server does not permit an operation, TIBCO ActiveMatrix BusinessWorks still allows you to configure the operations in the transaction. However, no exception is raised and the operations that are not supported by the application server are performed independent of the transaction.

If the transaction commits, all eligible activities in the transaction group commit. If the transaction rolls back, all eligible activities in the transaction group roll back. The transaction group commits automatically if all activities in the group complete and a non-error transition is taken out of the transaction group. If any errors occur while processing the activities in the group, even errors in activities that do not participate in the transaction, the transaction is rolled back and an error is returned. You should create an error transition out of the group to handle this situation.

Configuring JTA UserTransaction Transactions

To configure a JTA UserTransaction, select JTA UserTransaction as the transaction type of the group.

JDBC Connections used by JDBC activities in the transaction group must be configured to use JNDI as the connection type and specify the data source in the application server. Activities that use the JMS transport must be configured to use the XA connection factory in the application server. The JNDI configuration for JMS, EJB, or JDBC activities must specify the appropriate connection and JNDI Context information for the application server.

The JTA UserTransaction transaction group has the following fields on the transaction group’s Configuration tab:

Field	Description
Transaction Manager	A JTA UserTransaction shared configuration resource. See <i>TIBCO ActiveMatrix BusinessWorks Palette Reference</i> for more information about this resource.
Include Checkpoint	When this field is checked, an implicit checkpoint is performed after the last activity in the group completes successfully and before the transaction is committed. The deployment configuration must specify a database for storing process engine information and the JDBC Connection used must specify JNDI as the connection type for this checkpoint to participate in the transaction.
Checkpoint Duplicate Key	When the Include Checkpoints field is checked, you can specify a duplicate key in this field to perform duplicate checking. This is useful if the checkpoint included in the transaction is the first checkpoint in the process definition. If the process engine crashes after the checkpoint, restarted process instances can use the duplicate key to determine if the transaction has already committed. See Detecting Duplicate Process Instances on page 208 for more information about specifying duplicate keys.

XATransaction

The XA Transaction type allows you to specify an XA-compliant transaction manager provided by a third party that supports the interfaces `javax.transaction.TransactionManager` and `javax.transaction.Transaction`. The following XA-compliant transaction managers have been tested with TIBCO ActiveMatrix BusinessWorks:

- TIBCO BusinessWorks XA Transaction Manager 1.0
- Arjuna Transaction Service 4.0
- Java Open Transaction Manager (JOTM) 1.5.3 (this transaction manager does not support transaction recovery)

The XA Transaction type allows JDBC activities, ActiveEnterprise Adapter activities that use the JMS transport, and JMS activities to participate in transactions. Transactions across multiple databases are also supported. Also, if your process engine is configured to use a database for storage of process engine data, checkpoints can participate in an XA Transaction.



For JMS activities and ActiveEnterprise Adapter activities, request/reply operations cannot participate in an XA transaction.

Also, EJB activities cannot participate in an XA Transaction group.

The transaction manager is typically executed in the same Java VM as TIBCO ActiveMatrix BusinessWorks. Because of this, for third-party transaction managers, the path to the files needed to run the transaction manager must be placed in the TIBCO ActiveMatrix BusinessWorks classpath in the `bwengine.tra` and `designer.tra` files. For the TIBCO BusinessWorks XA Transaction Manager, configuration occurs during product installation.

If the transaction commits, all eligible activities in the transaction group commit. If the transaction rolls back, all eligible activities in the transaction group roll back. The transaction group commits automatically if all activities in the group complete and a non-error transition is taken out of the transaction group. If any errors occur while processing the activities in the group, even errors in activities that do not participate in the transaction, the transaction is rolled back and an error is returned. You should have an error transition out of the group to handle this situation.

Configuring XA Transactions

To configure an XA Transaction, select XA Transaction as the transaction type of the group.

JDBC Connections used by JDBC activities in the transaction group must be configured to use XA as the connection type. JMS activities must be configured to use the XA connection factory.

The XA Transaction group has the following fields on the transaction group's Configuration tab:

Field	Description
Transaction Manager	An XA Transaction shared configuration resource. See <i>TIBCO ActiveMatrix BusinessWorks Palette Reference</i> for more information about this resource.
Include In Transaction	Certain resources, such as JMS Queue Receiver, Wait for JMS Queue Message, or JMS Topic Subscriber expect acknowledge messages to be sent to the server. Normally, this is done by using a Confirm activity. These resources can also specify an acknowledge mode of "Transactional" that specifies their acknowledgement should be part of a transaction. This field allows you to specify the resource for which you would like to send an acknowledgement message. This message is part of the transaction, if the transaction completes successfully. There is no need for a Confirm activity for the resources specified in this field. Note: Transactional acknowledge mode is not supported for Wait for JMS Topic Message activities.
Include Checkpoint	When this field is checked, an implicit checkpoint is performed after the last activity in the group completes successfully and before the transaction is committed. The deployment configuration must specify a database for storing process engine information and the JDBC Connection used must specify XA as the connection type for this checkpoint to participate in the transaction.

Field	Description
Checkpoint Duplicate Key	When the Include Checkpoints field is checked, you can specify a duplicate key in this field to perform duplicate checking. This is useful if the checkpoint included in the transaction is the first checkpoint in the process definition. If the process engine crashes after the checkpoint, restarted process instances can use the duplicate key to determine if the transaction has already occurred. See Detecting Duplicate Process Instances on page 208 for more information about specifying duplicate keys.

Configuring Transaction Managers

When using a transaction manager with TIBCO ActiveMatrix BusinessWorks, you must configure the transaction manager and TIBCO ActiveMatrix BusinessWorks to work together properly. This section details the requirements for configuring the supported transaction managers.

TIBCO BusinessWorks XA Transaction Manager

The TIBCO BusinessWorks XA Transaction Manager is automatically configured for use with TIBCO ActiveMatrix BusinessWorks. However, in case TIBCO BusinessWorks XA Transaction Manager does not configure automatically, you can configure it manually.

To configure the BusinessWorks XA Transaction Manager manually, append the following paths to the `tibco.env.CUSTOM_EXT_APPEND_CP` property in `bwengine.tra` and `designer.tra` files:

```
<TIBCO_HOME>/bw/tm/1.0/lib
<TIBCO_HOME>/bw/tm/1.0/lib/ext
```

You may wish to set some of the custom engine properties available for this transaction manager. See *TIBCO ActiveMatrix BusinessWorks Administration* for more information about custom engine properties.

Java Open Transaction Manager (JOTM)

For Java Open Transaction Manager (JOTM), no changes are required to the JOTM environment to interact with TIBCO ActiveMatrix BusinessWorks. However, TIBCO ActiveMatrix BusinessWorks must be configured as follows to interact with JOTM.

1. The classpath for TIBCO Designer and TIBCO ActiveMatrix BusinessWorks must be modified to include the JOTM jar files. To accomplish this, the following JOTM directory must be included in classpath properties in the `designer.tra` and `bwengine.tra` files:

```
<JOTM_HOME>\jotm-1.5.3\config
<JOTM_HOME>\jotm-1.5.3\lib
```

2. The logging details for JOTM is specified in the TIBCO ActiveMatrix BusinessWorks `log4j` property file (`<BW_HOME>\5.12\lib\log4j.properties`). By default, the `log4j` property file is configured to send JOTM log messages to the TIBCO ActiveMatrix BusinessWorks log file (`BWLogFileAppender`).

To send JOTM log messages to a separate file, you must modify the TIBCO ActiveMatrix BusinessWorks `log4j` property file. There are several JOTM logger properties with the prefix, `log4j.logger.org.objectweb.jotm`. To each of these properties, add `jotm_log_file`.

See [Logging for Third-Party Components on page 215](#) for more information about the TIBCO ActiveMatrix BusinessWorks `log4j` file.

Arjuna Transaction Service

For the Arjuna Transaction Service (ATS), both the ATS environment and the TIBCO ActiveMatrix BusinessWorks environment must be modified.

The following configuration changes must be performed in the ATS properties file:

1. ATS provides two transaction manager implementations, one that is local (`com.arjuna.ats.internal.jta.transaction.arjunacore.TransactionManagerImpl`) and one that is based on the distributed JTA implementation (`com.arjuna.ats.internal.jta.transaction.jts.TransactionManagerImpl`). TIBCO ActiveMatrix BusinessWorks only supports the local implementations. For TIBCO ActiveMatrix BusinessWorks to work correctly, ATS must be configured to use the local implementations. Open the ATS properties file (`<ATS_HOME>\ats-4.0\etc\arjunajts-properties.xml`) and make sure that the properties `com.arjuna.ats.jta.jtaTMImpementation` and `com.arjuna.ats.jta.jtaUTImpementation` are set to the following values:

```
<property name="com.arjuna.ats.jta.jtaTMImpementation"
value="com.arjuna.ats.internal.jta.transaction.arjunacore.Transact
ionManagerImpl"/>
<property name="com.arjuna.ats.jta.jtaUTImpementation"
value="com.arjuna.ats.internal.jta.transaction.arjunacore.UserTran
sactionImpl"/>
```

2. ATS provides two recovery modules for XA resources, one for local transaction recovery and one for distributed transaction recovery. In TIBCO ActiveMatrix BusinessWorks, local transaction recovery is used. By default the ATS properties file may be configured to enable both the local transaction recovery (<property name="com.arjuna.ats.arjuna.recovery.recoveryExtension5" value="com.arjuna.ats.internal.jta.recovery.arjunacore.XARecoveryModule"/>) and distributed transaction recovery (<property name="com.arjuna.ats.arjuna.recovery.recoveryExtension6" value="com.arjuna.ats.internal.jta.recovery.jts.XARecoveryModule"/>).

TIBCO ActiveMatrix BusinessWorks uses the local transaction recovery, therefore the distributed transaction recovery property must be commented out or removed from the ATS properties file.

3. By default, the ATS ObjectStore is created in the current working directory of the application. When using TIBCO ActiveMatrix BusinessWorks, this will be the working directory or the directory where the TIBCO ActiveMatrix BusinessWorks process engine is started. It is recommended that you change the location where the ATS ObjectStore is created. To change the object store's location, set the value for the following property in the ATS properties file to your directory's location:

```
<property
name="com.arjuna.ats.arjuna.objectstore.objectStoreDir"
value="your directory location" />
```

The following configuration changes must be made to TIBCO ActiveMatrix BusinessWorks to interact with ATS:

1. The classpath for TIBCO Designer and TIBCO ActiveMatrix BusinessWorks must be modified to include the ATS property file and jar files. To accomplish this, the following directories must be included in classpath properties in the `designer.tra` and `bwengine.tra` files:

```
<ATS_HOME>\ats-4.0\etc
<ATS_HOME>\ats-4.0\lib
<ATS_HOME>\ats-4.0\lib\ext
```

2. The logging details for ATS is specified in the TIBCO ActiveMatrix BusinessWorks `log4j` property file (`<BW_HOME>\5.12\lib\log4j.properties`). By default, the `log4j` property file is configured to send ATS log messages to the TIBCO ActiveMatrix BusinessWorks log file (`BWLogFileAppender`).

To send ATS log messages to a separate file, you must modify the TIBCO ActiveMatrix BusinessWorks `log4j` property file. Comment out the line containing: `log4j.logger.com.arjuna=WARN, tibco_bw_log`. Remove the

comment from the line containing `#log4j.logger.com.arjuna=WARN, tibco_bw_log, ats_log_file`.

See [Logging for Third-Party Components on page 215](#) for more information about the TIBCO ActiveMatrix BusinessWorks log4j file.

Transaction Recovery

XA Transactions can be recovered in the event of a failure. This section details transaction recovery in the supported transaction managers.

TIBCO BusinessWorks XA Transaction Manager

By default, the transaction manager and the recovery manager are started in the same Java VM as TIBCO ActiveMatrix BusinessWorks. Therefore, the transaction recovery manager is started when TIBCO ActiveMatrix BusinessWorks is started, and no additional recovery configuration is necessary.

You may wish to run the recovery manager in a separate Java VM to allow transaction recovery to occur independently of TIBCO ActiveMatrix BusinessWorks. See the TIBCO BusinessWorks XA Transaction Manager documentation for more information about starting the recovery manager in a separate Java VM. No additional configuration in TIBCO ActiveMatrix BusinessWorks is necessary when running in this mode.



If you are using non-TIBCO JDBC drivers (such as Oracle or IBM JDBC drivers), then edit the file `<BW_HOME>/tm/1.0/bin/setupenv.bat` or `<BW_HOME>/tm/1.0/bin/setup_env.sh`. Set the variable `TIBCO_TM_JDBC_JARS` to the name of the non-TIBCO JDBC driver JAR file.

Java Open Transaction Manager (JOTM)

Transaction recovery is not supported by Java Open Transaction Manager (JOTM). Refer to the JOTM documentation for more information.

Arjuna Transaction Service

By default, both the Arjuna Transaction Manager and the Arjuna Transaction Recovery Manager are started in the same Java VM as TIBCO ActiveMatrix BusinessWorks. Therefore, the transaction recovery manager is started when TIBCO ActiveMatrix BusinessWorks is started, and no additional configuration is necessary. In this configuration however, the transaction recovery manager is disabled if TIBCO ActiveMatrix BusinessWorks is stopped or if the Java VM crashes, and transaction recovery only occurs when TIBCO ActiveMatrix BusinessWorks is restarted.

Running the Arjuna Transaction Recovery Manager in a separate Java VM allows transaction recovery to occur independently, regardless of the status of TIBCO ActiveMatrix BusinessWorks. See the ATS documentation for more information about starting the Arjuna Transaction Recovery Manager in a separate Java VM.

After a failure, the ATS ObjectStore may contain the serialized TIBCO ActiveMatrix BusinessWorks Java classes. When running the recovery manager in a separate Java VM, the recovery manager requires references to the TIBCO ActiveMatrix BusinessWorks .jar files to de-serialize the classes. Therefore, when starting the recovery manager in a separate Java VM, verify the following:

- The classpath for the Arjuna Transaction Recovery Manager must contain the location of the TIBCO ActiveMatrix BusinessWorks .jar files. The following libraries must be included in <ARJUNATS_HOME>\bin\setup-env.bat:

```
<BW_HOME>lib\palettes\plugins.jar
<BW_HOME>lib\engine.jar

<TRA_HOME>lib\TIBCOxml.jar
<TRA_HOME>lib\TIBCOrt.jar
<TRA_HOME>lib\TIBCrypt.jar
<TRA_HOME>lib\TIBCOjms.jar

<TPCL_HOME>jdbc\TIBbase.jar
<TPCL_HOME>jdbc\TIutil.jar
<TPCL_HOME>jdbc\TIOracle.jar
<TPCL_HOME>jdbc\TIsqlserver.jar
<TPCL_HOME>jdbc\TIsybase.jar
<TPCL_HOME>jdbc\TIdb2.jar

<J2EE_LIB>\jms.jar
<ORACLE_LIB>\ojdbc14.jar (when using an Oracle database)
```

- The ATS property file for the recovery manager must refer to the same ATS ObjectStore that is configured for TIBCO ActiveMatrix BusinessWorks (see [Configuring Transaction Managers on page 193](#) for a description of configuring the ATS object store).
- Do not allow more than one Arjuna Transaction Recovery Manager to refer to the same ATS ObjectStore. That is, do not start one recovery manager in the same Java VM as TIBCO ActiveMatrix BusinessWorks, and then start another recovery manager in a different Java VM when both recovery managers are pointing to the same ATS ObjectStore.
- Recovery may not be possible if the ATS ObjectStore is modified or destroyed. See the ATS documentation for more information.

Refer to the Arjuna Transaction Service documentation for more information about transaction recovery.



When using the Oracle database, the following exception is logged by the ATS recovery manager after invoking the method "getConnection" of the XAConnectionRecovery object:

```
"2004-08-24 17:20:56,929 [Thread-5] WARN com.arjuna.ats.jta.logging.loggerI18N
- [com.arjuna.ats.internal.jta.recovery.xarecovery1]
XARecoveryModule.xaRecovery got XA exception
javax.transaction.xa.XAException: [tibcosoftwareinc][Oracle JDBC
Driver][Oracle]ORA-00942: table or view does not exist,
XAException.XAER_RMERR"
```

This may be caused either by a missing table or by a lack of permissions.

Check that the database user which the recovery manager's connection is using is the same as the one the business logic is using. Also, check that this user has the correct permissions on relevant data dictionary tables and views for the XA system.

PreCreating XA Transactions At Event Sources

BusinessWorks supports including Event Sources such as JMS Queue Receiver, AE Subscriber in a transaction. For the XA Transaction Type, Event Sources can be included in the Transaction Group by selecting the Event Source in the Include In Transaction dropdown list.

The transaction is created when the message arrives at the Event Source. However, transactions are required to be created before the message arrives at the Event Source when IBM MQ Websphere is used. BusinessWorks enables this feature by providing an option to create transactions at the Event Sources before the message arrives.

Monitoring XA Transactions

TIBCO ActiveMatrix BusinessWorks provides TIBCO Hawk microagent methods for monitoring XA transactions processed by TIBCO XA Transaction Manager or Arjuna Transaction manager. You can use the Hawk console to return a list of transactions that are currently being processed, or you can return statistics for all transactions since the process engine started. The two microagent methods available for transaction monitoring are `ListInflightTransactions()` and `GetTransactionStatus()`. See *TIBCO Administrator User's Guide* for more information about these microagent methods.

JMS Local Transaction

The JMS Local transaction allows JMS activities to participate in a transaction.

The JMS specification defines the concept of a transacted JMS Session that can be used to transact sends and receives, all or none of which will get executed when the session is committed.

A session, when specified as transacted, supports a single series of transactions. Each transaction groups a set of produced messages and a set of consumed messages into an atomic unit of work. Multiple transactions organize the session's input message stream and output message stream into a series of atomic units. When a transaction commits, its atomic unit of input is acknowledged and the associated unit of output is sent. When a transaction rollback is done, all the produced messages (output stream) are destroyed and the consumed messages (input stream) are automatically recovered.

The JMS Local transaction type allows JMS sender activities - JMS Queue Sender and JMS Topic Publisher, Get JMS Queue Message activity, JMS Event Sources - JMS Queue Receiver and JMS Topic Subscriber, and Reply To JMS Message activities to participate in the transaction. At runtime, the underlying JMS activities use the same transacted JMS session to provide transaction semantics for messages sent and received by the JMS activities.

The ActiveEnterprise Adapter activities such as Publisher to Adapter, Adapter Subscriber and Adapter Request-Response Server can also participate in JMS Local transactions.



The JMS Requestors and Wait for JMS Queue/Topic Message activities will not participate in a local transaction even if they are included in a JMS Local transaction group.

The sessions participating in the JMS local transactions cannot process other messages outside the transaction. Since sessions are limited resources defined by the `Max Sessions` property, the number of messages that can be processed simultaneously, and hence the throughput, can be affected.

Configuring JMS Local Transactions

To configure a JMS Local transaction, select JMS Local Transaction as the transaction type of the group.

To include the JMS Receiver in JMS Local Transaction, choose the Event Source in Include In Transaction.

To exclude a JMS Send activity from a transaction, check the Override Transaction Behavior in the advanced tab.

Limitations while working with JMS Local Transactions

Several sessions within a single JMS Local Transaction group

Consider several JMS activities present within the same JMS Local Transactional group, but referring to different sessions. These activities may refer to

- different Shared JMS Connection resources

Although two Shared JMS Connection resources are exactly identical, but the name of the Event Source may differ. In such cases, they will not be a part of the same JMS Local Transaction

- different messaging domains such as Queues and Topics.

The JMS Local Transaction does not span across sessions. Each session is transacted individually and each session is individually committed in a single phase. However, when a failure occurs while committing a particular session, BusinessWorks stops the execution of the process with an exception and no attempt is made to rollback other sessions.

JMS 1.1 not supported

The JMS specification version 1.0.2b specifies that the messaging domains for Point-to-Point and Publish-Subscribe are handled by separate set of interfaces with no common base interface. Since BusinessWorks is compliant with this JMS Specification version, it is implied that Topic and Queue sessions cannot share the same JMS session, although they may be required to transact in a single session context. The Sessions created in JMS Local Transactions can be either QueueSessions or TopicSessions, but cannot be both. Thus places a limit on the activities that can participate in a single transacted session.

When activities using both Queue and Topic are in the same transaction group, it results in two separate transactions running independently. However, if one the transaction fails, it is difficult to predict the behavior of the other.

Potential deadlock in Request-Reply activity

Including Request-Reply activities in a transaction may result in a dead-lock situation when the Request and Reply happen on the same queue, and the Reply is bound to the message sent by the request.

Similarly, including JMS Sender and Get JMS Queue Message activities in the same transacted session with the same destination may result in a dead-lock.

Hence to avoid these potential deadlock situations, Request-Reply activities are not included in transactions.

Nested Transactions

Although it is technically possible to nest transaction groups, BusinessWorks does not support nesting transactions. Each group starts a separate transaction, irrespective of whether the group is nested or not. Transaction commits and rollbacks are always local to the nested group and are not affected by the outcome of the nesting group.

Nested Transactions

It is possible to place a transaction group in another transaction group, thus "nesting" transactions. However, each group starts a separate transaction and behaves independent of the nesting group. [Table 18](#) describes the effect of creating a transaction group in another transaction group.

Table 18 Effects of nesting transaction groups

Transaction type	JDBC	XA	JTA	JMS Local
JDBC	Nesting JDBC transactions in other JDBC transactions is allowed, but each transaction is independent. That is, each transaction commits or rolls back separately. The outcome of the parent transaction does not affect the nested transaction.	Nesting JDBC transactions in an XA transaction is allowed, but each transaction is independent.	Nesting JDBC transactions in a JTA transaction is allowed, but each transaction is independent.	Nesting JDBC transactions in a JMS Local transaction is allowed, but each transaction is independent.
XA	Nesting XA transactions in JDBC transactions is allowed, but each transaction commits or rolls back separately.	Nesting XA transactions in XA transactions is allowed, but each transaction commits or rolls back separately.	Nesting XA transactions in JTA transactions is allowed, but each transaction commits or rolls back separately.	Nesting XA transactions in JMS Local transactions is allowed, but each transaction commits or rolls back separately.
JTA	Nesting JTA transactions in JDBC transactions is allowed, but each transaction is independent.	Nesting JTA transactions in XA transactions is allowed, but each transaction is independent.	Nesting JTA transactions in JTA transactions is not allowed. An exception is thrown at the start of the nested transaction.	Nesting JTA transactions in JMS Local transactions is allowed, but each transaction is independent.

Table 18 Effects of nesting transaction groups

Transaction type	JDBC	XA	JTA	JMS Local
JMS Local	Nesting JMS Local transactions in JDBC transactions is allowed, but each transaction is independent.	Nesting JMS Local transactions in XA transactions is allowed, but each transaction is independent.	Nesting JMS Local transactions in JTA transactions is allowed, but each transaction is independent.	Nesting JMS Local transactions in other JMS Local transactions is allowed, but each transaction is independent. That is, each transaction commits or rolls back separately. The outcome of the parent transaction does not affect the nested transaction.

Summary of Transactions

The following tables summarize the support of each type of activity in each type of transaction.

Table 19 summarizes the JDBC transaction type.

Table 19 JDBC transaction type

Activity Type	Participate in JDBC Transaction Type?
JDBC activities	Yes
TIBCO iProcess BusinessWorks Connector activities	Yes
JMS activities	No
ActiveEnterprise Adapter activities (using the JMS transport)	No
EJB activities	No

Table 20 summarizes the XA transaction type.

Table 20 XA transaction type

Activity Type	Participate in XA Transaction Type?
Checkpoint (specified by the Include Checkpoint field of the transaction group)	Yes, when a database is used to store checkpoint data, and the JDBC Connection is configured to use the XA connection type.
JDBC activities	Yes, when the JDBC Connection is configured to use the XA connection type.
TIBCO iProcess BusinessWorks Connector activities	Yes, when an XA connection is used.
JMS activities ¹	Yes, when JMS is configured to use an XA connection

Table 20 XA transaction type

Activity Type	Participate in XA Transaction Type?
ActiveEnterprise Adapter activities (using the JMS transport) ¹ .	Yes, when JMS is configured to use an XA connection
EJB activities	No

1. Request/Reply operations (such as JMS Topic Requestor and JMS Queue Requestor) are not permitted in transactions. Process starters and Wait For... activities cannot be part of an XA Transaction group, but their acknowledge message can participate in the transaction, if the resource is configured to use the Transactional acknowledge mode and the resource is listed in the Include in Transaction field of the transaction group. However, the acknowledge message for Wait for JMS Topic Message activities cannot be part of the transaction.

Table 21 summarizes the JTA UserTransaction type.

Table 21 JTA UserTransaction type

Activity Type	Participate in JTA Transaction Type With the Following Application Server?		
	BEA	IBM	JBoss
Checkpoint (specified by the Include Checkpoint field of the transaction group)	Yes, when database used to store checkpoint data, and the JDBC Connection is configured to use the JNDI connection type.	Not supported	Not supported
JDBC activities	Yes, when the JDBC Connection is configured to use the JNDI connection type.	Not supported	Not supported

Table 21 JTA UserTransaction type

Activity Type	Participate in JTA Transaction Type With the Following Application Server?		
	BEA	IBM	JBoss
JMS activities ¹	No, when using TIBCO Enterprise Messaging Service. Yes, when using BEA JMS server.	Not supported	Not supported
ActiveEnterprise Adapter activities (using the JMS transport) ^{1.}	No, when using TIBCO Enterprise Messaging Service. Yes, when using BEA JMS server.	Not supported	Not supported
EJB activities	Yes	Yes	Yes

1. Request/Reply operations (such as JMS Topic Requestor and JMS Queue Requestor) are not permitted in transactions. Process starters and Wait For... activities cannot be part of a JTA UserTransaction group.

Chapter 12 **Process Instance Execution**

This chapter details aspects of process execution that you can control with configuration fields in the process definition.

Topics

- [Detecting Duplicate Process Instances, page 208](#)
- [Sequencing Process Instances, page 212](#)

Detecting Duplicate Process Instances

Duplicate messages should be detected and discarded to avoid processing the same event more than once. Duplicate detection is performed when a process instance executes its first Checkpoint activity. You must specify a value for the `duplicateKey` element in the Checkpoint activity input schema. This value should be some unique key contained in the event data that starts the process. For example, the `orderId` value is unique for all new orders.

The following describes the procedure for duplicate detection by the process engine:

1. An incoming message is received and a process instance is created.
2. Activities in the process instance are executed until the first Checkpoint activity is reached. The Checkpoint activity has a value specified for the `duplicateKey` input element.
3. The process engine checks the current list of `duplicateKey` values for a matching value.
 - a. If no process instance has stored the given `duplicateKey` value, the process engine stores the value and completes the Checkpoint activity.
 - b. If another process instance has already stored the given `duplicateKey` value, the process engine terminates the process and throws a `DuplicateException`.
4. Once a process engine stores a `duplicateKey` value and performs the Checkpoint for a process instance, no other Checkpoint activities in the process instance can be used to store a different `duplicateKey` value.

Using the algorithm described above, process engines can guarantee that no newly created or recovered process instances will execute if they have the same `duplicateKey` value. Therefore, you should take care in choosing the value of `duplicateKey` and ensure that it will be unique across all process instances.

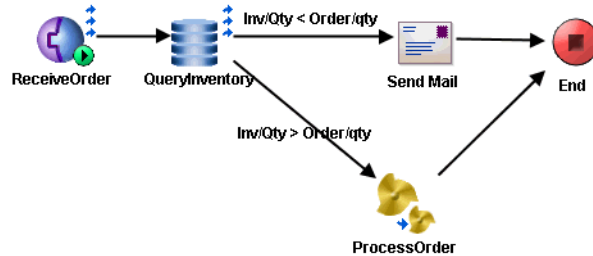


Duplicate detection can only be done across multiple engines on different machines if a database is used to store process engine data. If you are running fault tolerant process engines (that is, only one process engine is running at a particular time), or if all process engines run on the same machine, you can use a file system for process engine storage.

See *TIBCO ActiveMatrix BusinessWorks Administration* for more information on specifying process engine storage.

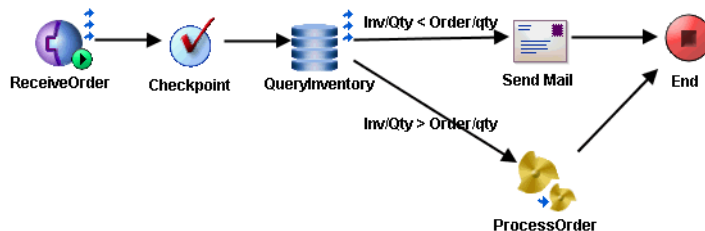
When to Perform Checkpoints

When detecting duplicate messages, it is important to place the Checkpoint activity before any activities that you do not want to execute more than once. For example, consider the following process definition.



In this process definition, an order is received, inventory is checked, and then either an email is sent to the inventory manager if the inventory is not sufficient, or the order is processed. In this example, you can either place the Checkpoint before the QueryInventory activity (because it is a database query and no actual change occurs) or after the activity but before either the Send Mail or ProcessOrder activities. It is a better choice to put the Checkpoint activity between the process starter and the QueryInventory activities.

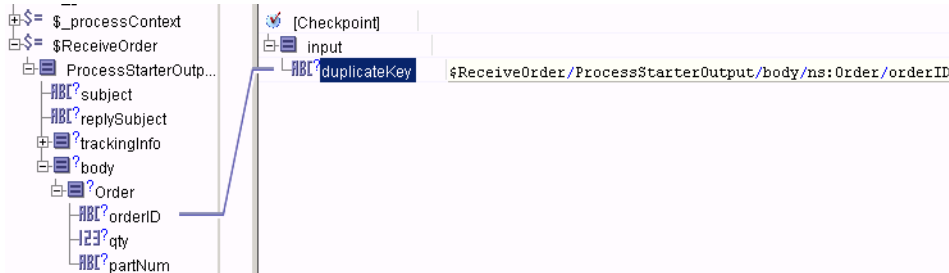
The following illustrates the example process definition with the Checkpoint activity properly placed.



Specifying the Duplicate Key

Duplicate detection is only as efficient as the `duplicateKey` that is specified. You should try to pick a value that is unique for every message. For example, you may select the `JMSMessageID` header property for JMS messages. In the example in the previous section, `orderId` is unique for each incoming order, so that would be a good choice for the value of the `duplicateKey`.

The following illustrates specifying orderID from the example above as the duplicateKey value in the Checkpoint activity.



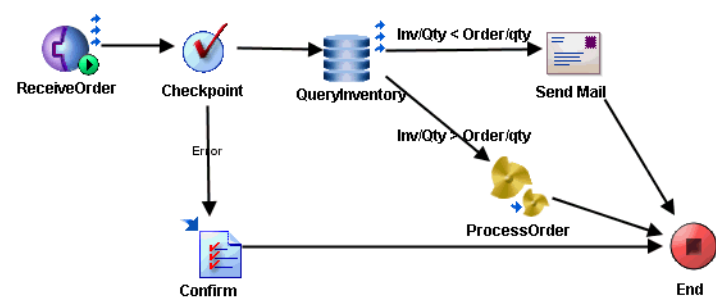
Transactions and Duplicate Detection

Transaction groups using certain transaction types can allow Checkpoint activities to be performed as part of the transaction. In this case, the checkpoint performed for the transaction may be the first checkpoint in the process definition. If this is the case, you can specify the duplicateKey as part of the transaction group configuration. The duplicateKey is specified in the Checkpoint Duplicate Detection Key field of the transaction group.

Handling Duplicate Messages

When a duplicate is detected, the Checkpoint activity fails with the DuplicateException. You can place an error transition from the Checkpoint activity to a series of activities to handle the duplicate message. If no error transition is specified, the process instance terminates and duplicate messages are effectively ignored.

The following illustrates an error transition added to the example process.



In this example, when a duplicate message is detected, the duplicate message is confirmed so that it is no longer redelivered, then the transition is taken to the end of the process definition.

Process Engine Properties for Duplicate Detection

The following process engine properties control duplicate key detection.

- `bw.engine.dupKey.enabled` — specifies whether duplicate detection is performed.
 `true` (the default) indicates the process engine will check for identical `duplicateKey` values.
 `false` indicates `duplicateKeys` when specified are ignored.
- `bw.engine.dupKey.timeout.minutes` — specifies how long (in minutes) to keep stored `duplicateKeys`. The default is 30 minutes.
 0 indicates the `duplicateKey` is removed when the job is removed. However, if `bw.engine.enableJobRecovery=true`, the job is not automatically removed after a failure so the `duplicateKey` will remain as long as the job remains. Such a job can be restarted or purged later.
 -1 indicates to store `duplicateKey` values indefinitely.
 Any positive integer greater than 0 indicates the number of minutes to keep stored `duplicateKeys`.
- `bw.engine.dupKey.pollPeriod.minutes` — specifies the number of minutes to wait before polling for expired `duplicateKey` values.

See *TIBCO ActiveMatrix BusinessWorks Administration* for more information about setting process engine properties.

Sequencing Process Instances

Process instances can be executed in the order they were created. TIBCO ActiveMatrix BusinessWorks allows you to specify a sequencing key on process starters that determines which process instances to execute sequentially. Process instances with sequencing keys that evaluate to the same value are executed in the order they were started. Process instances can have sequencing keys that evaluate to different values, but only process instances with the same value for the sequencing key are executed sequentially. Process instances with different sequencing key values can be executed concurrently.

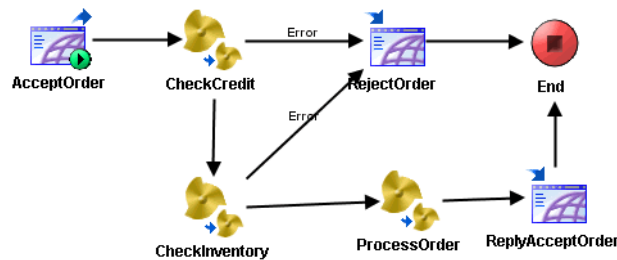
TIBCO ActiveMatrix BusinessWorks also allows you to control process execution administratively by setting properties in the deployment configuration of the project. TIBCO Administrator allows you to control the maximum number of process instances in memory as well as the maximum number of concurrently executing process instances. Using these settings, you can specify that all process instances should be executed sequentially in the order they were created. This method is not as flexible as using the Sequencing Key field on a process starter. Using the administrator settings is only recommended when you cannot change the process definitions in the project before deployment (for example, if you purchase a pre-built project from a third party).

This chapter details examples of using the Sequencing Key field on the Misc tab of process starters to control the order of process execution. For more information on using TIBCO Administrator to set deployment configuration parameters, see *TIBCO ActiveMatrix BusinessWorks Administration*.

Example 1: Processing Orders As They Are Received

In this example, an order-entry system must process incoming orders in the order in which they are received. The orders are placed using a web client and the TIBCO ActiveMatrix BusinessWorks process definition uses an HTTP Receiver process starter to accept the incoming orders. [Figure 38](#) illustrates the example process definition.

Figure 38 Example order-entry system

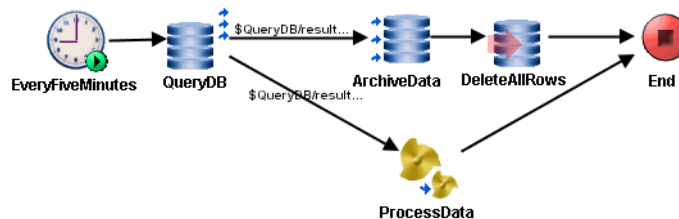


In this example, you wish to sequentially execute all orders that are accepted by this process definition. To accomplish this, specify a constant in the Sequencing Key field of the Misc tab of the AcceptOrder HTTP Receiver process starter. For example, place "OrderEntry" in the Sequencing Key field. Because you placed a constant in the field, the value of the XPath expression is identical for each incoming order. Therefore, all new orders are executed in the order they are received.

Example 2: Periodic Processing

In this example, the Timer process starter is used to start a process every five minutes. The process is a monitoring application that examines data in a database and performs the appropriate action. For example, if the number of rows returned by a query is greater than 5000, the process archives the data to another database and removes all rows from the table. If the number of rows is less than 5000, the results of the query are processed and stored in a separate table. [Figure 39](#) illustrates the example process definition.

Figure 39 Example of periodic processing



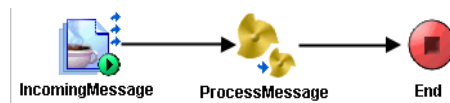
In this example, process instances are created every five minutes, but the data in the table changes much more frequently. You must ensure that the data returned by the query is processed before the next query executes. Therefore, you must execute each process instance sequentially before executing the next process instance. System backups occur at midnight each night, and there are no changes to the table during this time.

To configure sequential processing in this scenario, you can use the XPath expression `pfx:TimerOutputSchema/Day_Of_Month` in the Sequencing Key field of the Misc tab of the Timer process starter. This ensures that process instances created each day are completed sequentially in the order they are created. At the end of the day, no updates are occurring to the table, so process instances created at the end of each day can execute concurrently with the first process instance created the next day.

Example 3: Handling Client Messages

In this example, a JMS application sends messages to TIBCO ActiveMatrix BusinessWorks for processing. Each JMS client authenticates to the JMS server with a different user ID. Messages from different clients are permitted to be processed concurrently, but all messages received from the same client must be processed in the order they are received.

Figure 40 Example of handling incoming messages



In this example, you can use the expression `pfx:ActivityOutput/JMSProperties/pfx:JMSXUserID` in the Sequencing Key field of the Misc tab of the JMS Topic Subscriber process starter. This ensures that all messages from a specific client are processed in the order they are received.

Logging for Third-Party Components

TIBCO ActiveMatrix BusinessWorks can use a variety of third-party components. For example, the Apache Tomcat server is used to accept incoming HTTP or SOAP requests or the Arjuna Transaction Service can be used as a transaction manager. Many third-party components can use the standard log4j logging services. TIBCO ActiveMatrix BusinessWorks provides the `bw/<relNum>/lib/log4j.properties` file to allow you to configure logging services for third-party components. For more information about log4j, see <http://logging.apache.org/log4j/docs/>.

The properties defined in `bw/<relNum>/lib/log4j.properties` are required by the components used by TIBCO ActiveMatrix BusinessWorks. The supplied `log4j.properties` file has comments describing property usage. You can alter the properties in this file, if you wish to configure logging for your environment. Do not remove any required properties from this file.



It is a good idea to create a backup copy of the `log4j.properties` file before altering it. This allows you to return to the original configuration if your changes result in errors.

To print the location of the log4j configuration file used by log4j, set `java.property.log4j.debug` property to true.

There can be only one `log4j.properties` file per Java VM. If you wish to use properties from a different `log4j.properties` file, you can either add the properties to `bw/<relNum>/lib/log4j.properties` or you can alter the `bwengine.tra` file to point to the location of your own `log4j.properties` file.

If you use your own `log4j.properties` file, you must include all of the required properties from the file supplied with TIBCO ActiveMatrix BusinessWorks in your file.

Chapter 13 **Inter-Process Communication**

Executing process instances can communicate and can pass data between each other. The General Activities palette contains the Wait and Notify activities and the Receive Notification process starter for implementing inter-process communication. This chapter describes inter-process communication and provides examples of its use.

Topics

- [Overview of Inter-Process Communication, page 218](#)
- [Data for Inter-Process Communication, page 220](#)
- [Coordinating Inter-Process Communication, page 221](#)
- [Database Storage for Wait/Notify/Receive Notification Information, page 223](#)
- [Examples of Inter-Process Communication, page 224](#)

Overview of Inter-Process Communication

TIBCO ActiveMatrix BusinessWorks allows two executing process instances to communicate. You may need process instances to communicate if you wish to synchronize process execution or if your processes must execute in a specific order.

TIBCO ActiveMatrix BusinessWorks provides the Wait and Notify activities and the Receive Notification process starter to handle inter-process communication. These activities are similar to semaphores in programming. A process containing a Wait activity waits for another process to execute a corresponding Notify activity. Alternatively, the Receive Notification process starter creates a new process instance when another process executes the corresponding Notify activity.

The data sent between the activities is defined by a Notify Configuration shared configuration resource. The Notify activity only sends information to the Receive Notification process starter or Wait activity that specifies the same Notify Configuration resource. The schema supplied to the Notify Configuration resource can be empty, if you do not wish data to be sent between processes. However, the Notify Configuration resources must be the same for the Notify, Receive Notification, or Wait activities if they are to be used to communicate with each other.

A string-based key is used to coordinate between Wait/Notify/Receive Notification activities to determine when a Notify activity corresponds to a Wait or Receive Notification. The key is supplied to the Notify activity, and any Wait activity that matches that key is executed when the Notify occurs.

When a Notify activity executes its information is stored until a matching Receive Notification or Wait activity accepts the information. The Notify activity executes immediately and transitions to the next activity. The Notify activity cannot be used to determine when a corresponding Receive Notification or Wait has received the information.

In general, using inter-process communication consists of these steps:

1. Define the data that must be passed between the processes by creating a Notify Configuration shared configuration resource.
2. Determine the key that correlates processes with Notify activities with the corresponding processes with Receive Notification process starters or Wait activities.
3. Create process definitions that use the Receive Notification, Wait, and Notify activities. These activities are located in the General Activities palette. See

TIBCO ActiveMatrix BusinessWorks Palette Reference for more information about the configuration requirements for each of these activities.

4. If your process engines are on different machines, a database must be used to store process instance information. Wait/Notify information is stored in a database so that process engines on different machines can share information.

The following sections describe these steps in more details. See [Examples of Inter-Process Communication on page 224](#) for more specific examples of when inter-process communication is useful.

Data for Inter-Process Communication

The Notify Configuration shared configuration resource defines the data that the Notify activity passes to the corresponding Wait activity or Receive Notification process starter. The schema for the data is defined in the same way as any other schema is defined. See [Editor on page 56](#) for more information on specifying schemas.

The same Notify Configuration resource is used to configure the Notify activity as well as the Wait activity and the Receive Notification process starter. The schema in the Notify activity's configuration appears in the Notify activity's input schema. This allows you to map process variables to the Notify activity's input. The Notify activity then passes its data to its corresponding Wait or Receive Notification.

The Wait activity and Receive Notification process starter have output that matches the Notify Configuration specified on their Configuration tab. This allows you to use the data passed by the process with the Notify activity in subsequent activities after the Receive Notification or Wait activities.



If you wish only to signal the waiting process to continue but not exchange data, the Notify Configuration schema used by the Notify/Receive Notification/Wait activities can be empty. However, the same Notify Configuration resource must be specified by corresponding Notify and Receive Notification or Wait activities. Only activities with the same Notify Configuration resource can communicate with each other.

Coordinating Inter-Process Communication

When configuring Receive Notification, Wait, and Notify activities, you must specify a key to coordinate which activities correspond to each other. You can also specify a timeout for how long to keep the information about Wait and Notify activities before it is removed from storage. The following sections describe configuring the key and timeouts for inter-process communication.

Specifying the Key

To configure Receive Notification, Wait, and Notify activities, you must specify a key that correlates Notify activities with Receive Notification or Wait activities. The key is a string that corresponding activities specify to determine when a Receive Notification or Wait activity should accept data from a Notify activity. The key is similar to event keys used in activities that wait for incoming events (described in [Event on page 54](#)). The key is a string, but you can use any XPath expression that evaluates to a string when the process instance executes.



XPath expressions can be used to specify the key for Wait and Notify activities. The Receive Notification process starter can specify a global variable or a fixed string for its key.

Each Notify activity corresponds to exactly one Receive Notification or Wait activity. That is, as a Notify activity executes, the first Receive Notification or Wait activity that matches the Notify's key can then execute. You can execute many Notify activities with the same key.

You can create one-to-one correspondence between Wait and Notify activities so that exactly one process' Notify activity corresponds to one other process' Receive Notification or Wait activity. Or, you can create many-to-one relationships so that many Notify activities' keys can correspond to the Receive Notification or Wait in one process. A Notify, however, always only corresponds to only one Receive Notification or Wait activity. Therefore, once a Notify executes, the corresponding Receive Notification or Wait activity continues processing.

See [Examples of Inter-Process Communication on page 224](#) for examples of specifying keys for Wait/Receive Notification/Notify activities.

Timeouts for Notify and Wait

Notify and Wait activities have associated timeouts. Timeouts specify how long information for the Notify and Wait is kept before it is removed from storage.

The Notify activity executes immediately and transitions to the next activity in the process definition. However, the timeout value for the Notify activity specifies how long the notification information should be kept. If no corresponding Wait activity executes before the specified timeout, the information is removed. Once notification information is removed from storage, it cannot be accepted by the corresponding Wait activity.

The Wait activity causes process execution to pause until a corresponding Notify activity with a matching key executes. The Notify activity can execute before the corresponding Wait activity and its information is then waiting in storage. If the Notify has not executed, the process instance containing the Wait suspends until the Notify occurs or the Wait activity's specified timeout is reached.

The Receive Notification process starter does not have a timeout because it creates a process instance only when a corresponding Notify activity executes.

Database Storage for Wait/Notify/Receive Notification Information

If your process engines are located on different machines, a database is required to store process instance state for inter-process communication. When process engines reside on different machines, they must share information about pending Wait/Receive Notification/Notify activities.

A database allows process engines to share process instance state information across machines in a domain. Your deployment configuration must specify a database for process engine storage if you expect process instances on different machines to have corresponding Wait/Receive Notification/Notify activities.

See *TIBCO ActiveMatrix BusinessWorks Administration* for more information on specifying a database for process engine storage when configuring your deployment.

Examples of Inter-Process Communication

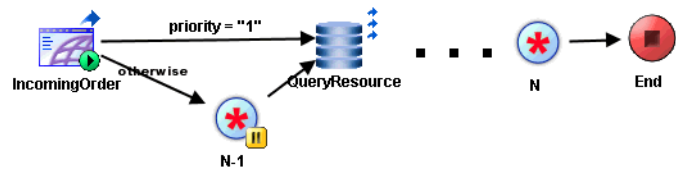
The following sections describe situations where inter-process communication may be necessary.

Enforcing Order for Process Execution

If you wish process instances to execute in a certain order, you can use Wait and Notify activities to accomplish this. This requires using some portion of the incoming event to determine the order.

For example, an application may assign a priority number to incoming orders. This may be important when accepting requests for a limited resource, for example, airplane seats or available space in a university class. The orders must be processed in order of priority to ensure that orders with the highest priority number have access to the resource first. [Figure 41](#) illustrates an example process definition that orders incoming events.

Figure 41 Ordering incoming events



In this process definition, new requests are submitted through a web interface. A new process is started for each request, and a priority number (an ordered sequence) is given with each request. The order with priority number 1 is submitted, and processed immediately. When the first order is completed, a Notify is sent with its key set to 1. All other orders transition to the Wait activity. These orders are suspended until a Notify activity is executed whose key is equal to their priority number minus one (that is, the order with the next highest priority number).

Using this technique, orders are processed in the order of their priority, regardless of when the order is submitted. All orders create a process instance and then immediately suspend until the notification is sent from the order with the next highest priority.

Multiple Types of Incoming Events Resume a Running Process

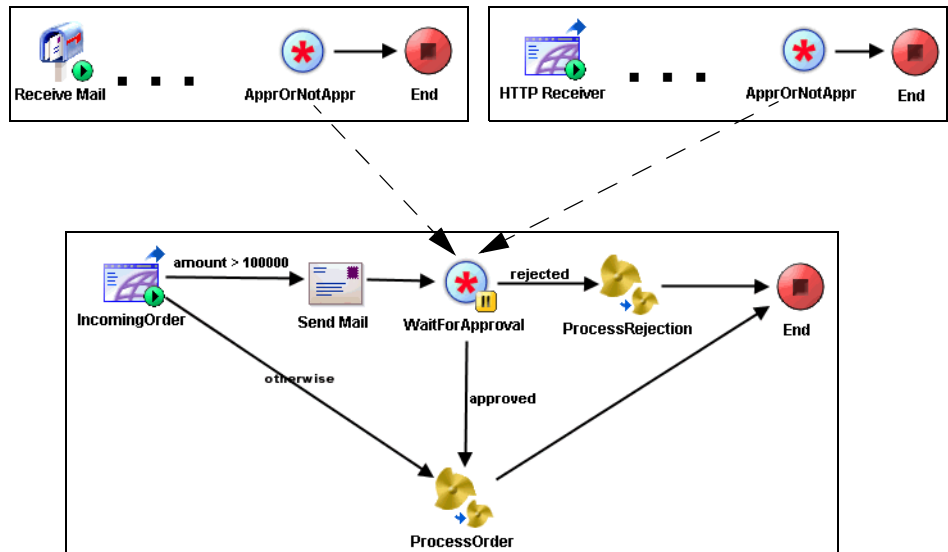
Some processes require a "Wait for..." event (for example, Wait for Adapter Message) to continue processing. This occurs when the process requires an external application to send an additional request.

For example, a new order arrives, and because the total is over \$100,000, it requires approval before processing. You may notify a group of approval managers by email, then any of the approval managers can respond by email or through a web interface for approval. [Figure 42](#) illustrates this set of process definitions.

Figure 42 Multiple event sources to continue a process

Orders can be approved by mail or by web.

Whichever Notify occurs first is accepted by the Wait



Incoming Order process handles orders less than \$100,000 immediately.

The Wait/Notify activities use the OrderID as the key to determine the order that corresponds to the notification. In this case, it is possible for more Notify activities to execute than Wait activities. You must configure the Notify activities to have an appropriate timeout so that the notify information is removed if it is not used by the associated Wait activity.

Scalability With Incoming Events

You may distribute incoming events across multiple process engines. This allows for greater scalability because the load of incoming events is distributed. However, "Wait for..." activities, such as Wait for Rendezvous Message are configured on static filters and cannot be configured to filter based on dynamic data. A "Wait for..." activity starts listening from the time the engine is loaded. When you have a "Wait for ..." activity that uses reliable delivery in your process definition, the incoming event is received by all process instances across the multiple process engines. This can potentially affect the performance because of greater network traffic, depending upon how many process instances are running.

Process definitions with "Wait for..." activities running in a multi-engine mode discard the events when an event is not routed to the correct process and may result in the loss of events.

Ideally, you should create some mechanism so that incoming events are handled outside of the process definition and then routed to only the correct process definition. The Wait and Notify activities can accomplish this. You would replace your "Wait for..." activity with a Wait activity. Then, create a new process definition that contains a process starter to handle the incoming event. Use the Notify activity to send the data from the incoming event to the correct process instance with the corresponding Wait activity.

Specific Protocol Requirements

Some business processes use protocols with specific requirements that make inter-process communication necessary. For example, you may have a process that starts when a TIBCO Rendezvous Certified Message (RVCM) arrives. Your process may also require a Wait for Rendezvous Message listening on the same subject as the process starter. This specific configuration is difficult to implement because incoming messages create new processes and also are sent to the waiting activities in the process.

In the example above, the business requirements necessitate working around the requirements of the RVCM protocol. To accomplish this, you may be able to change your business requirements, or you can use the Wait and Notify activities to create two process definitions.

The first process definition accepts all new messages and determines, based on message content, whether the message should start a new process or be passed to an activity in the process waiting for the message. The process executes a Notify activity for the new message, but the key of the Notify is different depending upon whether a new process must start or if the message is to be sent to an executing process. The second process definition starts with a Receive

Notification process starter and has a Wait activity in place of the Wait for Rendezvous Message activity. This configuration allows the first process to receive all incoming messages, parse them to determine the appropriate action, and then pass each message to the appropriate process.

Chapter 14 **Invoking and Implementing Web Services**

Web services provide a standards-based approach for application integration between or within organizations. This chapter describes implementing and invoking web services using TIBCO ActiveMatrix BusinessWorks.

Topics

- [Overview of Web Services, page 230](#)
- [The Service Palette, page 232](#)
- [The SOAP Palette, page 245](#)
- [Web Service Wizards, page 260](#)
- [Using Web Services Security Policies, page 264](#)
- [WSIL Files and UDDI Registries, page 274](#)

Overview of Web Services

An organization can offer its application services to other organization by using the standards-based model of web services. Web services can be used within a business to integrate various critical applications, or web services can be made available to other businesses or individuals.

Web services rely on a variety of published standards for communication, security, data exchange, and so on. Standards-based technology enables businesses and individuals to use each other's web services, regardless of the underlying applications or implementations of the service. Most web services comply with the following standards:

- HyperText Transfer Protocol (HTTP) or Java Message Service (JMS) — HTTP and JMS are transports, that is mechanisms for relaying information. Web services use a transport mechanism such as a Web server or JMS server to store and relay messages. TIBCO ActiveMatrix BusinessWorks supports HTTP and JMS as transports for web service messages.
- Simple Object Access Protocol (SOAP) version 1.1 or 1.2 — SOAP is the communications protocol for web services. SOAP defines message structure and bindings to the underlying transports.
- EXtensible Markup Language (XML) — XML is used to define data schemas for SOAP message content.
- Web Service Definition Language (WSDL) — WSDL describes the interface to a web service. A web service provider publishes a WSDL file that describes the offered service. A client uses the WSDL file to determine the appropriate input, output, and fault messages for the service.
- Web Service Security (WS-Security) — WS-Security specification defines the standards-based approach to message-level security. Unlike transport-level session security (such as HTTPS), message-level security allows you to secure messages that may travel through multiple hops on a distributed transport channel. Message-level security is important for organizations that require trusted, secure communication between web services and clients.

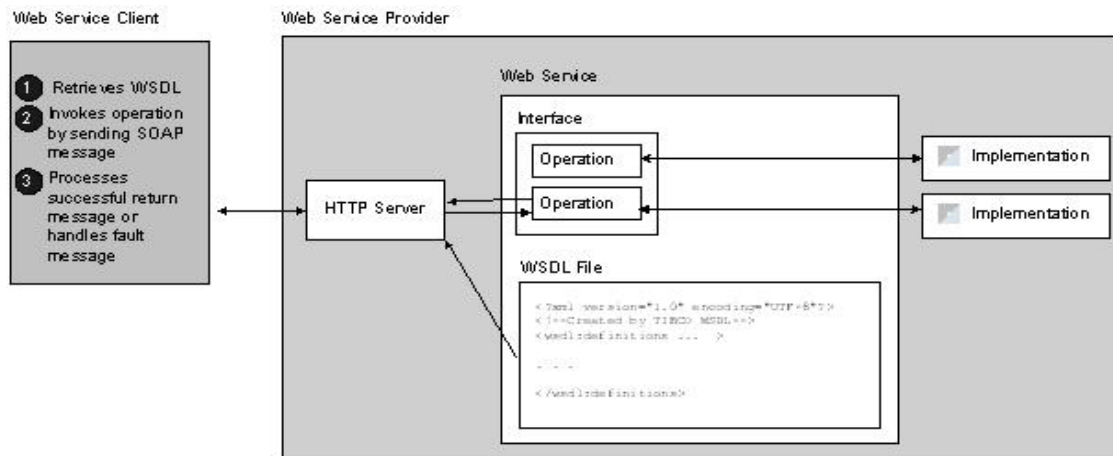
The World Wide Web Consortium (W3C) maintains the standards upon which web services are based. See <http://www.w3.org/> for more information about the currently supported web services standards.

Figure 43 illustrates a typical interaction between a web service client and server. In this example, the web service provider uses HTTP as the underlying transport for sending and receiving messages. The client first retrieves the interface for the web service by requesting the WSDL file. Retrieving the WSDL file can be done

either through a direct request to the service provider (as illustrated) or by searching well-known directories of web services. See [WSIL Files and UDDI Registries on page 274](#) for more information about using directories of web service providers.

Next, the client invokes an operation in the service by sending a SOAP message with the appropriate input defined in the WSDL. The web service executes the appropriate implementation and either successfully completes (and optionally sends a message containing the results of the operation) or sends a fault message detailing any errors encountered during the operation. Operations can be one-way (no information returned) or request-response (a response message is returned). If the operation returns information, the client then processes the information.

Figure 43 A typical interaction between web service client and server



TIBCO ActiveMatrix BusinessWorks can act as either a web service client or a web service provider. Process definitions can act as web service clients and invoke web service operations by using the activities in the SOAP palette. See [The SOAP Palette on page 245](#) for more information about how to invoke web services within a process definition.

There are two approaches to implementing web services as a web service provider in TIBCO ActiveMatrix BusinessWorks. The SOAP Event Source process starter creates a simple service with one operation over one transport. The Service resource allows you to create a service that implements multiple operations offered over multiple transports. See [The Service Palette on page 232](#) and [The SOAP Palette on page 245](#) for more information about using TIBCO ActiveMatrix BusinessWorks as a web service provider.

The Service Palette

A service implements one or more interfaces and exposes one or more endpoints per interface. In web service terminology, the interface would be a PortType in a WSDL 1.1 file. The endpoint would be referred to as a port. However, the term endpoint does not imply a physical network address where the service is listening. Services provide an abstraction of web services so that you can implement a more generic service-oriented architecture (SOA) that is not dependent upon a specific transport or implementation. The Service palette contains resources that allow you to define services.

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.

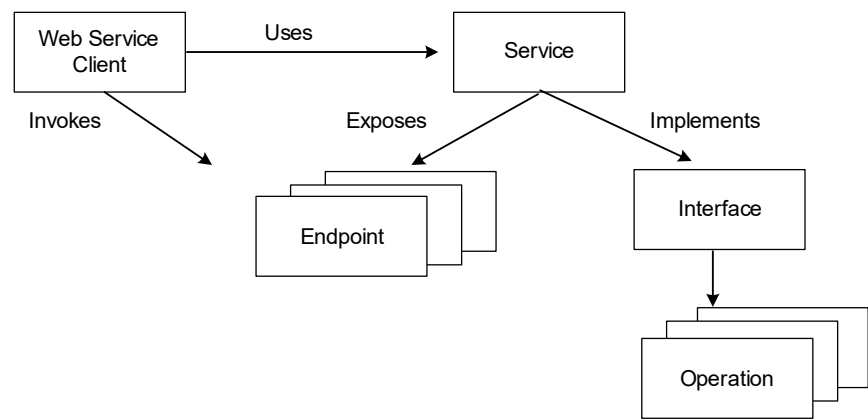
In this release of TIBCO ActiveMatrix BusinessWorks, each service can implement any number of interfaces. The endpoints available are SOAP over HTTP, SOAP over JMS and Local.

For example, you may wish to offer a bidding service that implements a Buyer interface. The interface is exposed on the SOAP over HTTP endpoint. The operations contained in the Buyer interface are Bid, Retract, and Raise.

See *TIBCO ActiveMatrix BusinessWorks Palette Reference* for a detailed description of the resources in the Service palette.

Figure 44 illustrates the service model.

Figure 44 Service model



A service joins an abstract WSDL interface to concrete implementations and exposes them on one or more endpoints. A web service client invokes one of the abstract interface operations using configuration settings provided by the service in the form of concrete WSDL bindings.

Context Separation

Because the Service resource separates the service definition from the transport and the implementation of operations, your implementations may require context information from the underlying request on the specific transport. For example, you may wish to alter processing of the request based on the user ID of the requestor, or you may wish to access the client's digital signature. The Context resource allows you to define a schema for storing relevant context information.

You can define Context resources, each containing the schema for the data you wish to use. Each operation within a Service resource can then reference a particular Context resource. You can map data from the incoming request (for example, MIME message attachments, authentication and SSL information, and so on) into the elements of the schema defined in your Context resource.

When a Context resource is specified, the data from the incoming request is passed to the implementation defined for an operation. The process definition that implements the operation can use the Get Context and Set Context activities to retrieve or set values for elements within the context.

See *TIBCO ActiveMatrix BusinessWorks Palette Reference* for more information about the Context resource and the Get Context and Set Context activities.

Configuring a Service

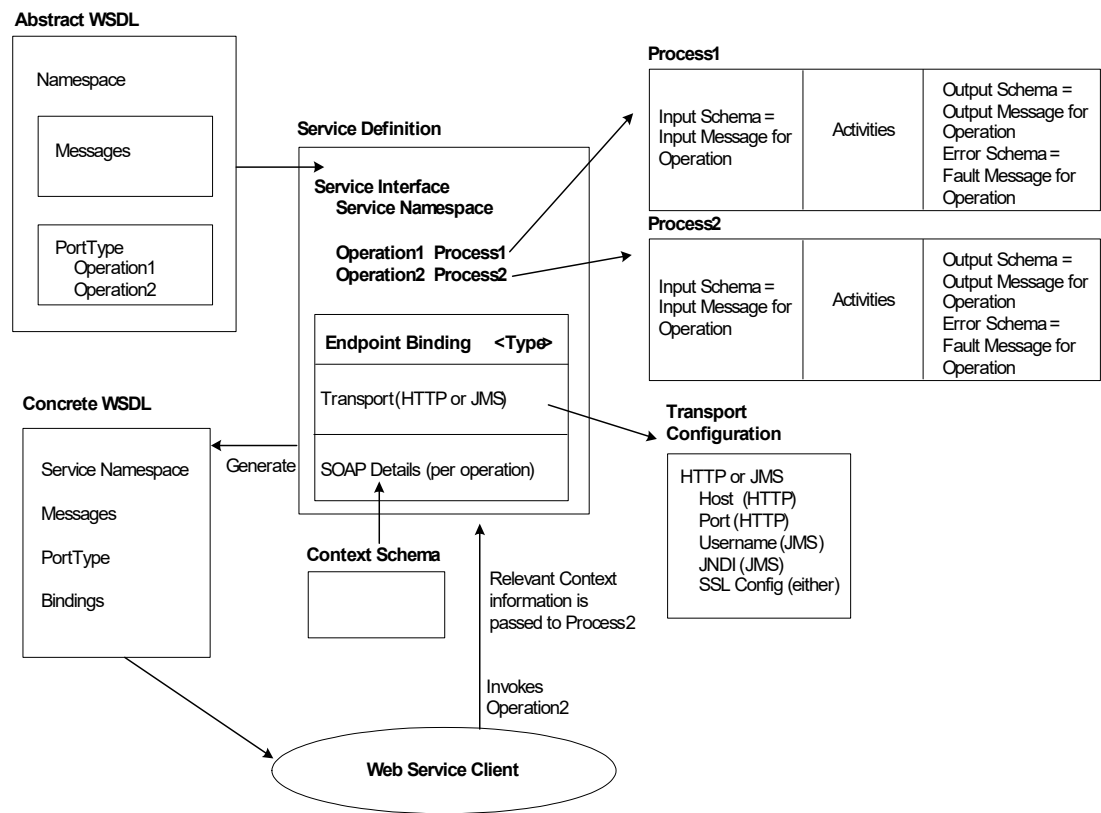
When specifying a Service resource, you must supply the following:

- a WSDL file — this defines the interface for the service. This interface describes the operations contained in the service and input, output, and fault messages for the operations.
- endpoint bindings — endpoints expose the location of the service to the service partner. An endpoint binding specifies the transport to use for the endpoint, any transport-specific properties, and any SOAP details (such as SOAP version, SOAP action, and so on).
- operation implementation — each operation must specify the process definition to execute for the operation.

You can also specify other items, such as headers for input and output messages and context information from the client's request.

Figure 45 illustrates the relationship between the configuration elements of a service and the web service client. The Service palette chapter of *TIBCO ActiveMatrix BusinessWorks Palette Reference* describes how to configure a Service resource more completely.

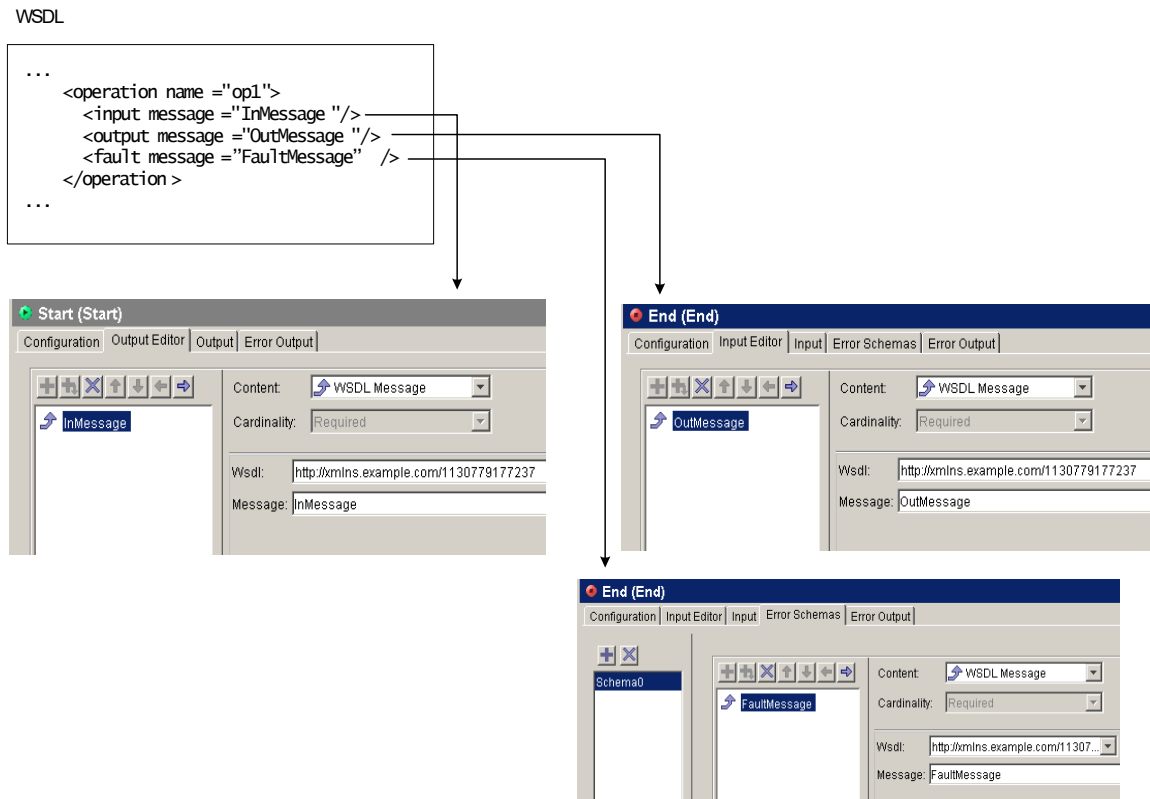
Figure 45 Relationship between WSDL, Service, Process, and web service clients



Operation Implementation

In TIBCO ActiveMatrix BusinessWorks, process definitions implement operations. The input, output, and error schemas for the process definition that implements an operation must match the input, output, and fault messages specified for the operation. Figure 46 illustrates the WSDL file and matching process input, output, and error schema specifications for a process definition. See [Start Activity on page 67](#) and [End Activity on page 69](#) for more information about specifying input, output, and error schemas for process definitions.

Figure 46 Input, output, and fault messages of operation must match respective schemas of process



You can specify only one WSDL message for the input schema, or output schema, or for each error schema. This message must be the root of the schema and the schema should not contain any other messages or schema elements. Do not specify more than one WSDL message or any other schema elements.

Running Services

When you deploy and run a project containing a Service resource, a service agent is created for the Service resource. The service agent waits for incoming requests on its specified endpoints and creates process instances to execute the implementation of the requested operation.

For each incoming request, the service agent passes the request's input message and any specified context information to the process instance that implements the requested operation. The process instance executes and sets the appropriate values in the output message or fault message, if an error occurs. The service agent then sends the output or fault message to the client.

Partners and Partner Link Configurations

Partners are external services that can be invoked from BusinessWorks Process Definitions. Partners are defined by a name and a WSDL portType that describes the operations that can be invoked. The Invoke Partner activity is used within a process definition to invoke an operation on a partner service. The Receive Partner Notification activity is used within a process definition to invoke notification services, where the notification service sends a message that the invoker receives. Partners can be located either inside the same project as the process that invokes the service, or the partner can be an external partner service invoked over the Internet by way of the SOAP protocol.

To define a partner for the process definition, follow this general procedure:

1. Use the + and X buttons to add or delete partners in the table in the Partners field for a process definition. 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 process definition is used as an operation implementation. If you do not specify the partner link here, you must specify the partner binding on the service.

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.

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

To create a partner link configuration resource, follow this general procedure:

1. Drag and drop a Partner Link Configuration resource 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.

Applying WS-Security Policy for Partner Link Configuration

Endpoint operation of a partner link defined in a Partner Link Configuration resource acts as a security subject and the security policies can be applied to it. There is a unique association between the partner link service endpoint operation security subject and the applied security policies.

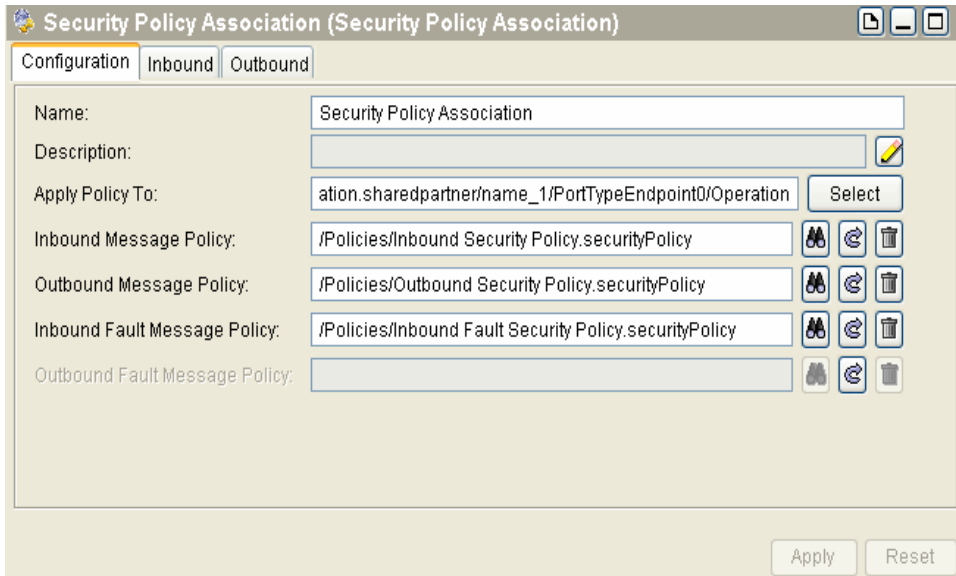
The SOAP event source policies are applied at the operation level. The policies applied to a Partner Link Configuration resource are always at the service endpoint operation level.

To apply a policy on a Partner Link Configuration, follow this procedure:

1. Drag and drop a Partner Link Configuration resource onto the design panel.
2. Click the + button to add Partner Link with **SOAPEndpoint**. If necessary, use the X button to delete partner links or the arrow buttons to move partner links up or down the list.
3. Drag and drop Security Policy Association resource onto the design panel.
4. Drag and drop Security Policy resource onto the design panel.

- 5. Name the Security Policy, for example, Inbound Security Policy.
- 6. From the **Policy Type** drop-down, select **inbound**.
Follow the similar process of outbound, inbound fault and outbound fault.
- 7. Click the **Select** button to the right of the **Apply Policy to** field. In the **Choose Security Subject** dialog box, expand Partner Link Configuration resource and select a service endpoint operation.
- 8. Click  in the **Inbound Message Policy** field. In the **Select a Resource** dialog box, expand **Policies** and select, Inbound Security Policy, in this case.
Follow the same process for Outbound Message Policy, Inbound Fault Message Policy, and Outbound Fault Policy.

Figure 47 Security Policy Association



- **Apply Policy To** - for associating the security policy with the PortTypeEndpoint.
- **Inbound Message Policy** - for applying security policy to the messages being received.
- **Outbound Message Policy** - for applying security policy to the messages being sent.
- **Inbound Fault Message Policy** - for applying security policy to the fault messages being received.

- **Outbound Fault Message Policy** - for applying security policy to the fault messages being sent.

Associating Policies with the Invoke Partner Activity

Invoke Partner activities are bound using the partner links and support the message level security.

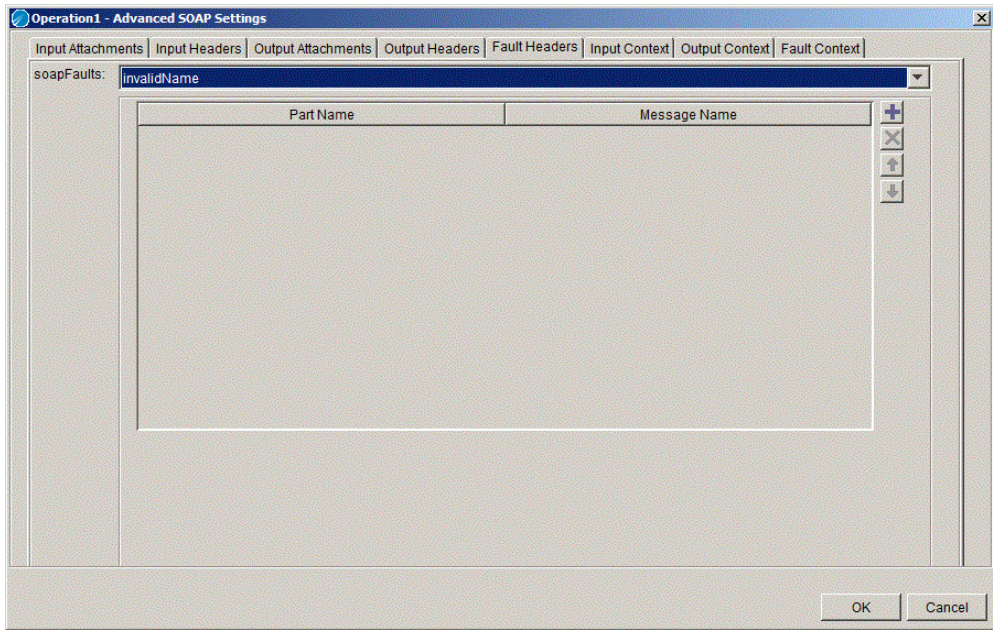
Using the service endpoint operation as the security subject ensures consistency in associating the security policies with the service resources. The service resource also exposes the service endpoint operation as a security subject. Exposing partner links as security policy subjects ensures a WS-Security support for all the TIBCO ActiveMatrix BusinessWorks or TIBCO ActiveMatrix BusinessWorks BPML Expansion constructs that use partner links for outbound invocations. This implies that the Invoke Partner activity is WS-Security enabled if, it is bound using a partner link that has a SOAP endpoint.

If a user associates the security policies to a particular partner link service endpoint operation, the WS-Security processing is enabled and performed according to the WS-Security guidelines. All the supported WS-Security constructs must work in context with the partner link service endpoint operation security subject.

Headers for Declared Faults

This feature enables the user to configure headers for declared faults. With this the user will be able to configure different schemas to be mapped to the headers for different Fault messages. A **Fault Headers** tab (Figure 48) has been added to the **Advanced SOAP Settings**. You can map the context resource values to the configured header using the **Fault Headers** tab.

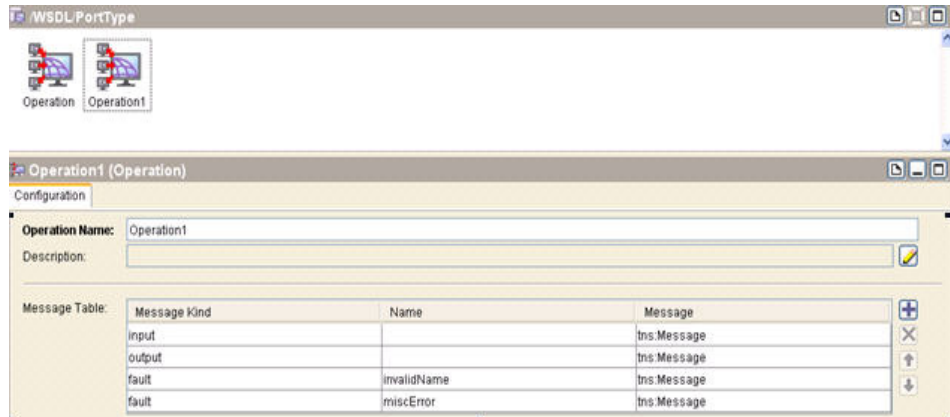
Figure 48 Fault Headers



Configuring Headers for the Fault Message

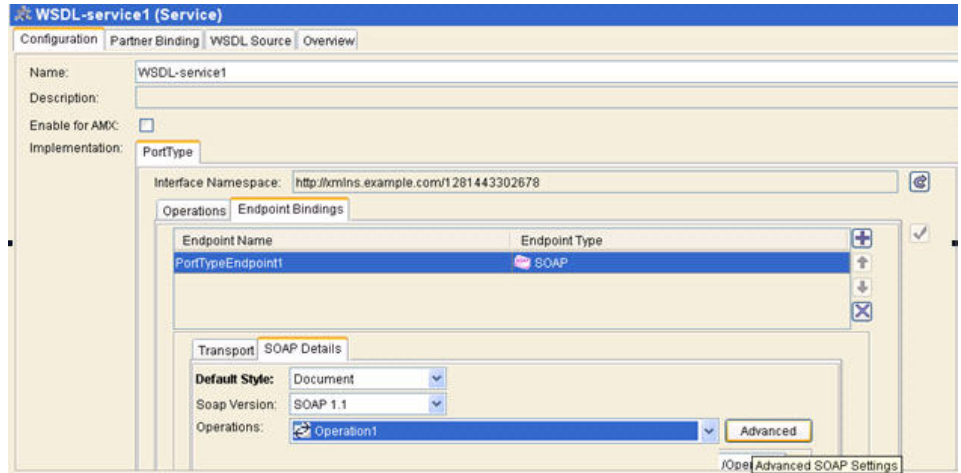
To configure headers for a fault message, as shown in [Figure 49](#), for example, configure headers in the context resource in the same way they were configured in the headers for declared Fault:

Figure 49 Faults Configured in the Service Resource



1. Click "Operation1" to go to the **Advanced** tab of the Service Resource.

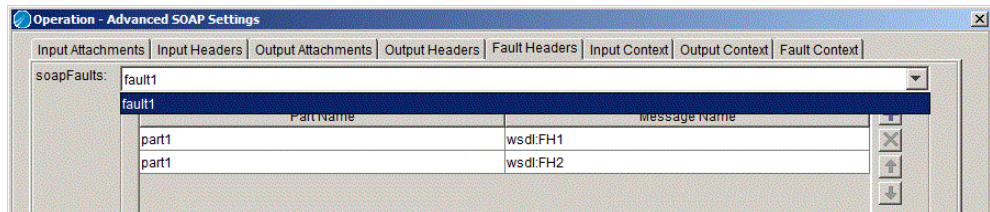
Figure 50 Advanced Tab for Configured Fault Message



2. Click the Fault Headers tab.

Both the Faults available for that operation are listed in the soapFaults drop-down as shown in Figure 51. Configure the fault message for any of the selected fault. Select the fault header name and Partname of the selected fault same as the Output Headers.

Figure 51 SOAPFault Messages



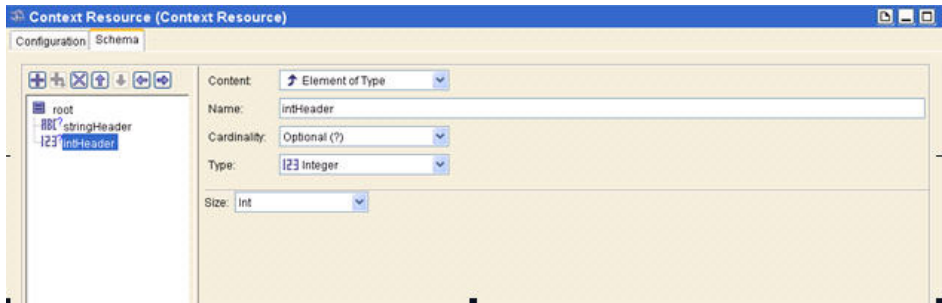
Configuring Context Resource

The way headers are configured in [Headers for Declared Faults](#), define the Context Resource that can be used to map user inputs to the configured headers.



Irrespective of the number of fault messages for which headers are configured, there will be only one shared context resource to be used. Hence, the schema defined for this context resource will be a *superset* of all schemas.

Figure 52 Context Resource

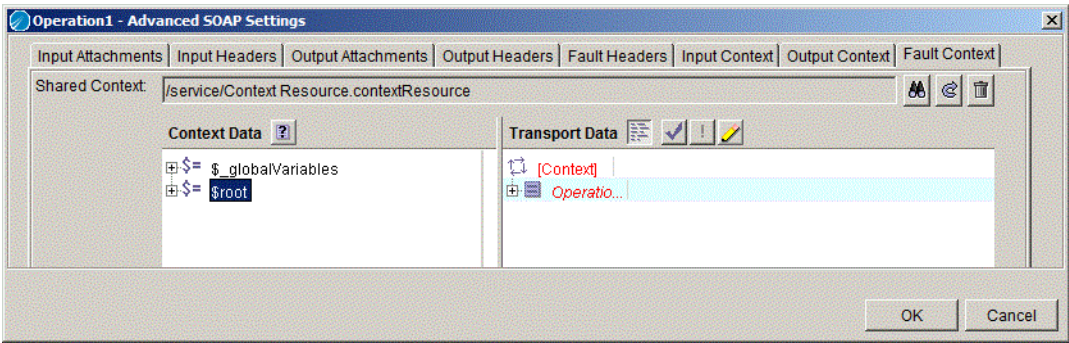


Mappings on the Fault Context Tab

To do mappings on the Context Resource:

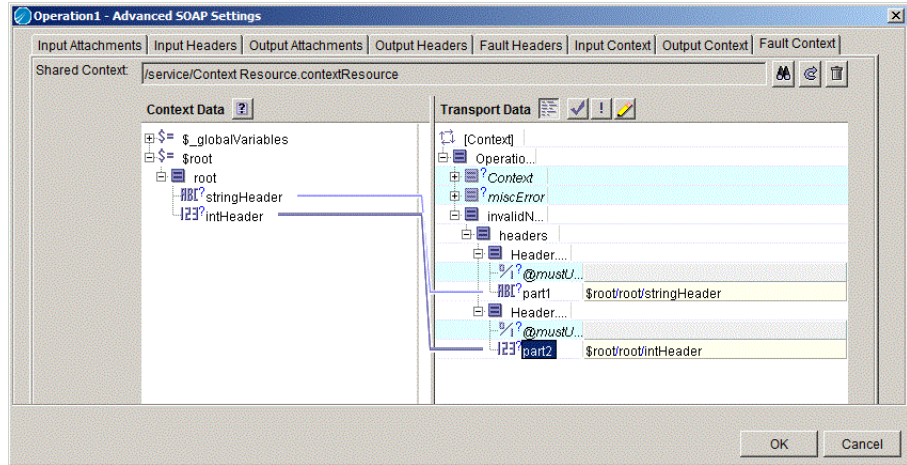
1. Go to Fault Context tab and pick the configured context resource.

Figure 53 Mappings on Fault Context



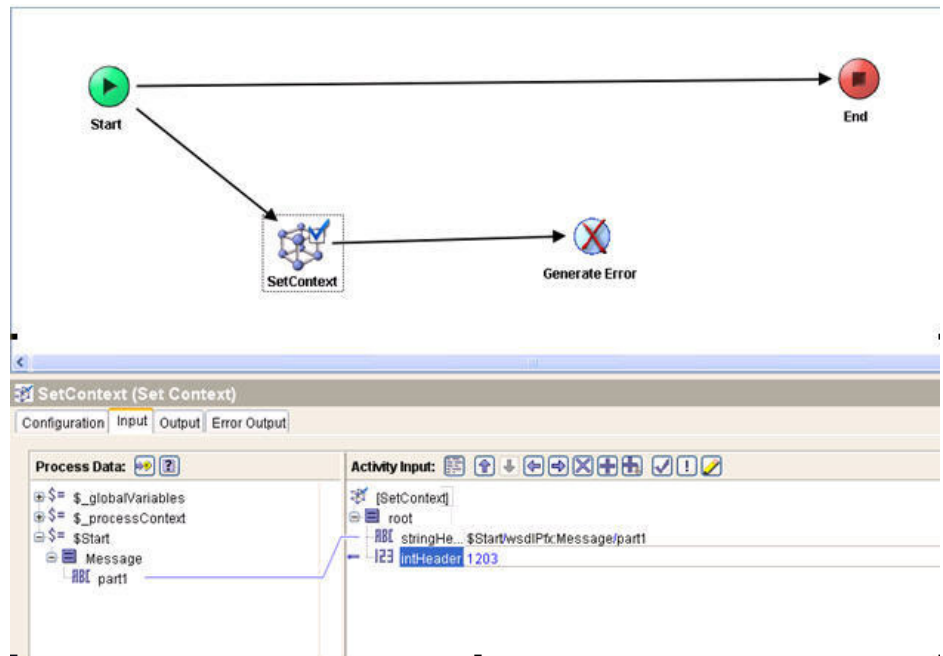
2. Map the Context Data to the configured headers in the Transport Data.

Figure 54 Mapping Configured Headers



In the Implementation process before sending the `invalidName` fault, use the `Set Context` Activity and configure values as shown in Figure 55.

Figure 55 Implementation Process





Ensure that right header context is set before sending the appropriate fault message.

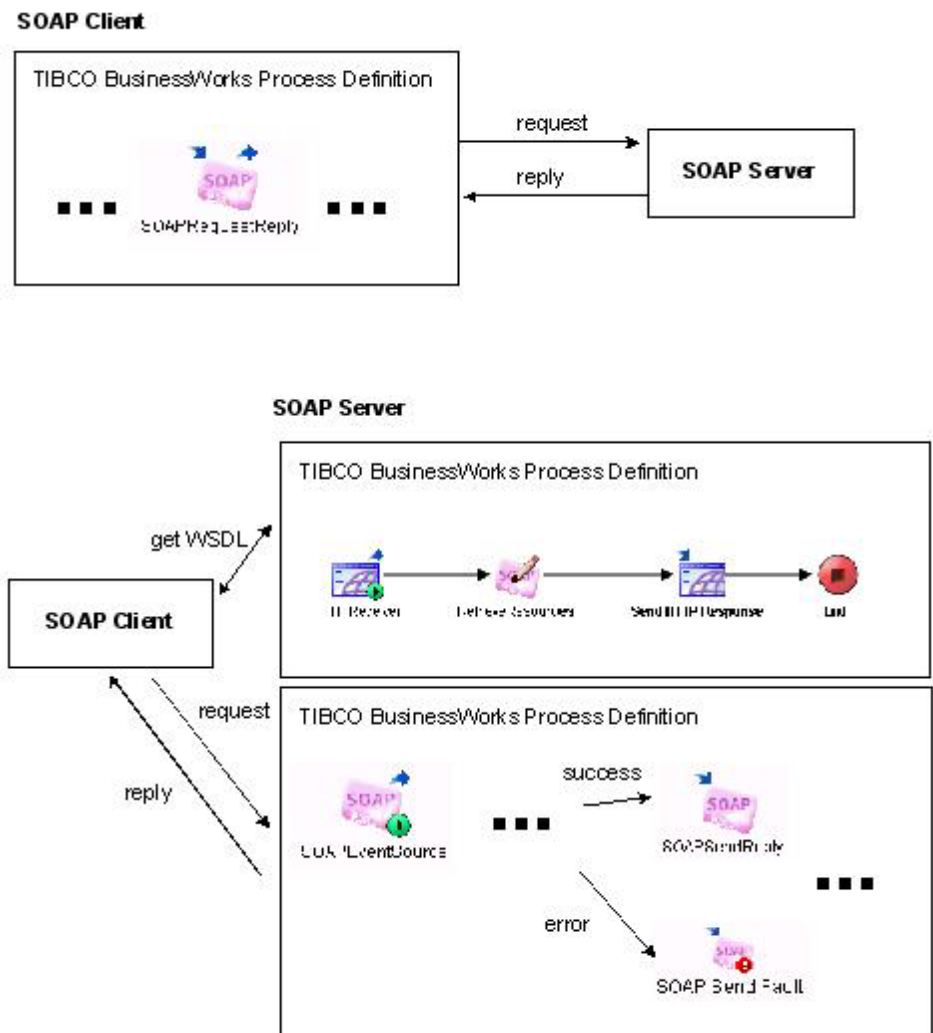
Configuring fault in the declared fault shows the Fault Header on the wire in the outgoing SOAP fault message and incoming SOAP fault message under the header section.

For example, if you configure headers for `invalidName` fault and use the `SetContext` activity to configure them in the process but select `throw miscError` fault in the `Generate Error` activity, then that fault header configuration will be ignored.

The SOAP Palette

TIBCO ActiveMatrix BusinessWorks can act as a SOAP client and send SOAP requests to remote web services, or TIBCO ActiveMatrix BusinessWorks can act as a SOAP server and handle incoming SOAP requests. Figure 56 illustrates how the activities of the SOAP palette allow TIBCO ActiveMatrix BusinessWorks to send and receive SOAP requests.

Figure 56 Activities in the SOAP palette



Sending SOAP Requests

A TIBCO ActiveMatrix BusinessWorks process definition can perform a SOAP request operation using the SOAP Request Reply activity. In this case, the process acts as a SOAP client that sends a request to a web service. The web service is described in a WSDL file. You can retrieve WSDL files from well-known web service registries. See [WSIL Files and UDDI Registries on page 274](#) for more information.

Once a WSDL file is retrieved, you can store and use the WSDL file in the project. You can view and manipulate the WSDL file with the WSDL palette. See *TIBCO ActiveMatrix BusinessWorks Palette Reference* for more information.

The web service may be configured to send a reply or a fault. The SOAP Request Reply activity executes synchronously. When the web service sends a fault, the fault is stored in the error output of the activity and an error transition is taken to the next activity in the process definition. One-way operations do not receive reply or fault messages.

Receiving SOAP Requests

TIBCO ActiveMatrix BusinessWorks can implement a web service through a process definition. The SOAP Event Source process starter defines the concrete WSDL file for the web service. SOAP clients must retrieve the concrete WSDL file before sending a SOAP request to the service. The WSDL Palette allows you to locate and store existing concrete service definitions or create your own abstract service definition. If you use an existing concrete service definition, the concrete bindings are ignored, and you can configure the service to use different namespaces and input/output headers.



Using the SOAP Event Source process starter allows you to use WSDL interface files that have only one operation. That is, you can only implement one operation in the process definition that has the SOAP Event Source. If you wish to implement a service with multiple operations, use the Service resource.

To create a web service, perform the following general procedure:

1. Create XSDs to define the input, output, and fault messages for the web service.
2. Create a WSDL File resource that defines the interface to the web service.
3. Create a process definition that performs the work of the web service. Use the SOAP Event Source process starter for this process definition. Use the SOAP Send Reply activity to return a reply or the SOAP Send Fault activity to return a fault to the client.

4. Create a process definition that retrieves the concrete WSDL file and sends it to web service clients. Use the HTTP Receiver process starter and the Retrieve Resources activity for this process definition.

See *TIBCO ActiveMatrix BusinessWorks Palette Reference* for more information about the resources in the SOAP palette.

Processing SOAP Messages with Attachments

SOAP messages can have message parts that contain attachments. For a message part to contain an attachment, specify the Special Type in the Type field of the Part Details section of the Message resource when creating a WSDL file configuration. See *TIBCO ActiveMatrix BusinessWorks Palette Reference* for information about creating WSDL files.

Messages with Attachments

SOAP clients that send messages with attachments must conform to the SOAP Messages with Attachments specification (<http://www.w3.org/TR/SOAP-attachments>).

As described in this specification, the messages are transmitted separately as a MIME attachment and the SOAP message may or may not have a reference to the attachment. The actual data for attachments is always contained within the `mimeEnvelopeElement` of the output schema for the given resource. This element contains a repeating element named `mimePart` that holds the list of attachments in the SOAP message. The attachment list can be correlated to the message parts containing the attachments by using the `content-id` `mimeHeader` element. Note that the attachment is not a part of the SOAP infoset.

For all existing projects, the SwA attachment style is the default option.

Message Transmission Optimization Mechanism

Message Transmission Optimization Mechanism (MTOM) provides another way of sending binary content or attachment processing by serializing SOAP messages with attachments. This is available only with SOAP version 1.2. Optimization is only available for element content that is in a canonical lexical representation of `xs:base64Binary` data type. Message Transmission Optimization Mechanism (MTOM) conforms to the specification <http://www.w3.org/TR/soap12-mtom>.

For an outbound SOAP Message with MTOM attachments sent by SOAP Request Reply or SOAP Send Reply activities, any element of type `xs:base64Binary` (or an extension of `xs:base64Binary`) in the SOAP response message is treated as an MTOM attachment and appears as a separate MIME part on the wire.

For an inbound SOAP Message with MTOM attachments received by the SOAP Event Source or SOAP Request Reply activities, the SOAP message will have a XOP include reference to the attachment. Any XOP include reference in the SOAP message will be replaced by the corresponding attachment's content encoded in Base64. Any MIME attachment that is not referenced from the SOAP Message using XOP include reference will be ignored by the MTOM Processing layer.

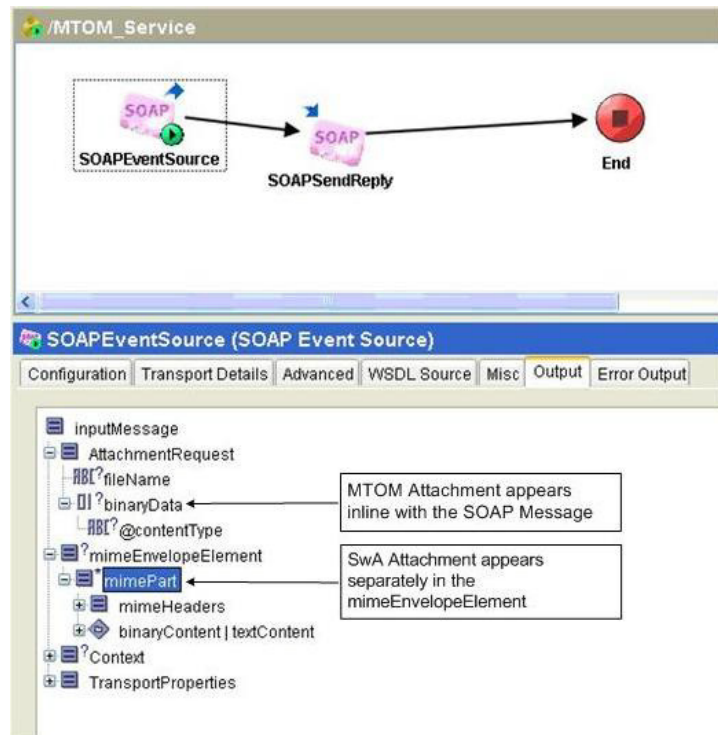
Following is an example of how a MTOM message is transmitted on the wire and the XOP mechanism is used to reference the attachments from the SOAP message.

```
MIME-Version: 1.0
Content-Type: Multipart/Related;boundary=MIME_boundary;
    type="application/xop+xml";
    start="<mymessage.xml@example.org>";
    startinfo="application/soap+xml; action=\"ProcessData\""
Content-Description: A SOAP message with my pic and sig in it
--MIME_boundary
Content-Type: application/xop+xml;
    charset=UTF-8;
    type="application/soap+xml; action=\"ProcessData\""
Content-Transfer-Encoding: 8bit
Content-ID: <mymessage.xml@example.org>
<soap:Envelope
    xmlns:soap='http://www.w3.org/2003/05/soap-envelope'
    xmlns:xlmime='http://www.w3.org/2004/11/xlmime'>
  <soap:Body>
    <m:data xmlns:m='http://example.org/stuff'>
      <m:photo
        xlmime:contentType='image/png'><xop:Include
          xmlns:xop='http://www.w3.org/2004/08/xop/include'
          href='cid:http://example.org/me.png' /></m:photo>
        <m:sig
          xlmime:contentType='application/pkcs7-signature'><xop:Include
            xmlns:xop='http://www.w3.org/2004/08/xop/include'
            href='cid:http://example.org/my.hsh' /></m:sig>
        </m:data>
      </soap:Body>
    </soap:Envelope>
  --MIME_boundary
Content-Type: image/png
Content-Transfer-Encoding: binary
Content-ID: <http://example.org/me.png>
```

```
// binary octets for png
--MIME_boundary
Content-Type: application/pkcs7-signature
Content-Transfer-Encoding: binary
Content-ID: <http://example.org/my.hsh>
// binary octets for signature
--MIME_boundary--
```

Although the attachment is transmitted outside the SOAP message with a reference to it, the attachment appears as if it's embedded in the SOAP message. The binary content or the attachment is a part of the SOAP Infoset. MTOM-style attachments are available inline with the SOAP message as shown in [Figure 57](#).

Figure 57 MTOM Attachment



As a result, the attachment threshold configuration is not applicable to MTOM attachments.



A runtime error will be thrown if a SOAP 1.1 message with MTOM attachments is received.



A way to handle two specific non-standard MTOM/XOP message scenarios:

Scenario 1: Non-standard content-type header value in root MIME part in the payload.

Set property 'java.property.com.tibco.net.mime.mpwob true' in `designer.tra` or `bwengine.tra`.

"mpwob" stands for "Multi-parts without boundary". As per MIME multipart spec MIME multipart message without a specified boundary is illegal. Setting this property to 'true' provides a mechanism by which a Content-Type header with a multipart field, but without a boundary field, will be treated as a body part.

Without this property, the MIME parser will throw an exception when it encounters such a non-standard compliant Content-Type header.

Scenario 2: XML-Invalid Characters in SOAP Envelope.

The SOAP Envelope in the payload had characters which are reserved characters in XML and cannot be used in XML content.

For example, a Windows file name path, `C:\.`

To get around this issue, turn off validation for the Service Resource or SOAP Event Source.

- To turn off Validation for all Job creators, set the following property to "false":

```
java.property.Validate.JC
```

- To turn off validation for a particular Service Resource, set the following property to "false":

```
java.property.Validate.JC.<service resource name>.serviceagent
```

- If the Service Resource is inside nested folders, then set the property to "false" using the following pattern property:

```
java.property.Validate .JC.<folder1>/<folder2>/<service  
resource name>.serviceagent
```

Handling Large Attachments

TIBCO ActiveMatrix BusinessWorks supports large attachments using the Write to File feature. The support for large attachments is added to the SOAP, Service and HTTP activities (SOAP Request Reply, SOAP Event Source, SOAP Send Reply, Service Resource, Send HTTP Request, HTTP Receiver, and Send HTTP

Response). These large attachments can be the files of gigabyte size which are shared between the Client and Server irrespective of the Heap size. For details, refer to the Write to File section in the respective activities in *TIBCO ActiveMatrix BusinessWorks Palette Reference*.



Service resource handles only large incoming attachments. It does not support large outgoing attachments.

Ensure that the Threshold Data Size is less than the Heap Size and select the "Exclude File Content" checkbox in the 'Read File' activity. You must also map the 'fileName' output of Read File activity to the 'fileName' attribute of the respective activities' mimePart.



Write to File is now supported for both SwA and MTOM attachments.

SOAP Undescribed Headers

TIBCO ActiveMatrix BusinessWorks SOAP header support is added on the Service Resource to handle Undescribed headers. Undescribed headers are SOAP headers that may appear in the SOAP message but are not part of the WSDL binding description. For **Inputs**, their content has to be mapped into a context object to make it available to the implementing process, as these headers are not a part of the WSDL message. For **Output**, they have to be mapped to the SOAP message from a context object, as these headers are not part of the WSDL binding.

Undescribed header configuration is available only for SOAP endpoints in the Service resource. This support is not available for SOAP Event Source Activity.

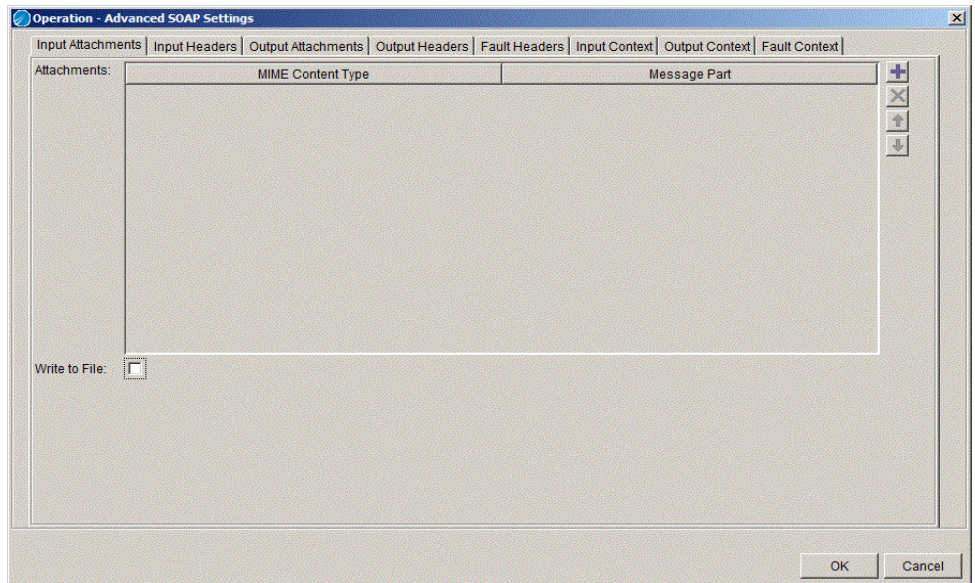


Recommended. WSDL Messages intended for Undescribed Headers should ideally have unique part names. If this is not followed, there can be ambiguity in resolving SOAP headers and there may be ambiguity in the Context Mapping.

Perform the following general procedure before defining the Undescribed Header:

1. Create a WSDL Service.
2. In the Configuration tab, click **EndPoint Bindings** and select **PortType endpoint**.
3. Click **Advanced** button. This will open the **Advanced SOAP Settings** screen.

Figure 58 Advanced SOAP Settings

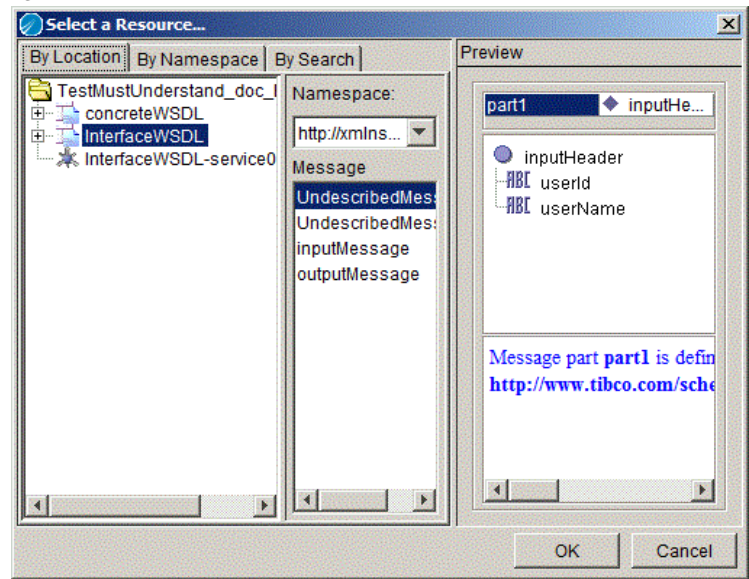


To define Undescribed Header, perform the following general procedure:

These steps can be followed for Output Context as well. Undescribed headers are available for both Input and Output Context.

1. Click **Input Headers** tab. This will show a table with Message name and Part name. Use the + and X buttons to add or delete message in the table. Move the messages up or down the list using the arrow buttons.
2. Click  in the Message name field and click WSDL. Select Undescribed message and click OK as shown in [Figure 59](#).

Figure 59 Select a Resource

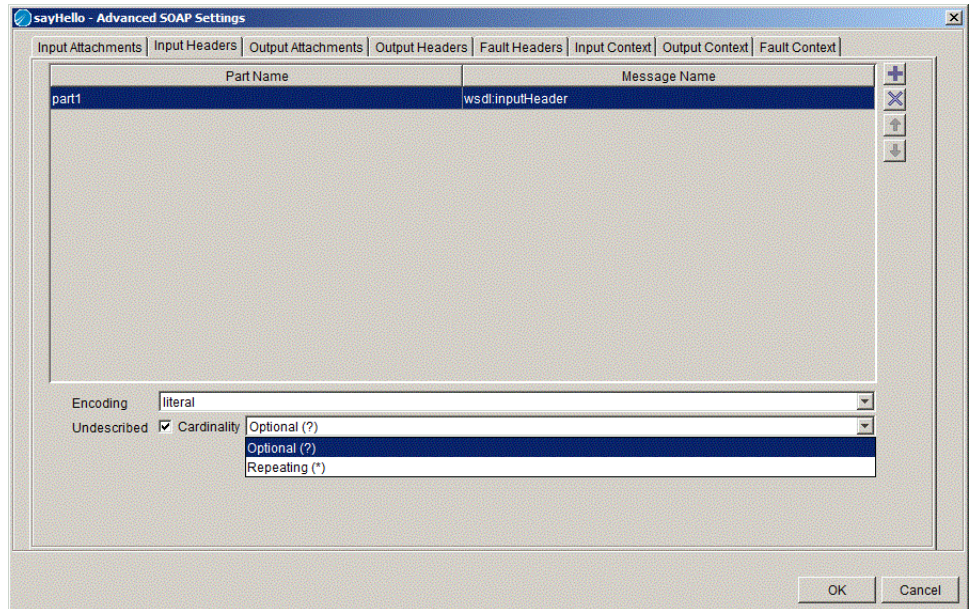


3. Select the Undescribed message.
4. Select the Cardinality from the drop-down as shown in Figure 60. Click OK.
The cardinality of the header element will depend on the cardinality chosen by the user.



Cardinality cannot be specified for Described headers. Cardinality for described headers is always 1.

Figure 60 Select Cardinality



Undescribed headers will be available for context mapping in Input Context and Output Context tab, depending on the configuration.



For context to be available for Output Context mapping at run-time, you must set the context using a SetContext activity.

When an InvokePartner activity in the implementing process is invoking a service referenced by the Partner Link, make sure to use the same Context resource in the 'Service Resource Input Context mapping' and in the 'Partner Link Configuration Output Context mapping'.

A GetContext must be performed before using the InvokePartner activity which overwrites the context resource provided as service input. As a best practice, you should use different Context Resources for different context mappings.

Partner Link Configuration for Undescribed Header Support

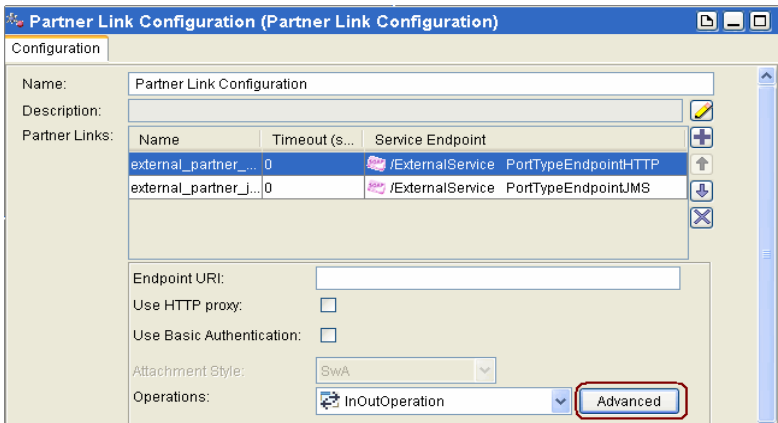
Undescribed headers are SOAP headers may appear in the SOAP message but are not part of the WSDL binding description. For Operation Input, as these headers are not part of the binding, they must be mapped to the SOAP message from a context object. For Operation Outputs, as these headers are not part of the WSDL message, their content must be mapped into a context object to make it available to the invoking process.

For a Partner Link Configuration, all defined headers are compulsorily undescribed because the described headers are defined in the concrete WSDL by the Service provider.

Undescribed headers are available for configuration only on SOAP endpoints.

1. Click **Partner Link Configuration** and in the **Configuration** tab click **Advanced** button.

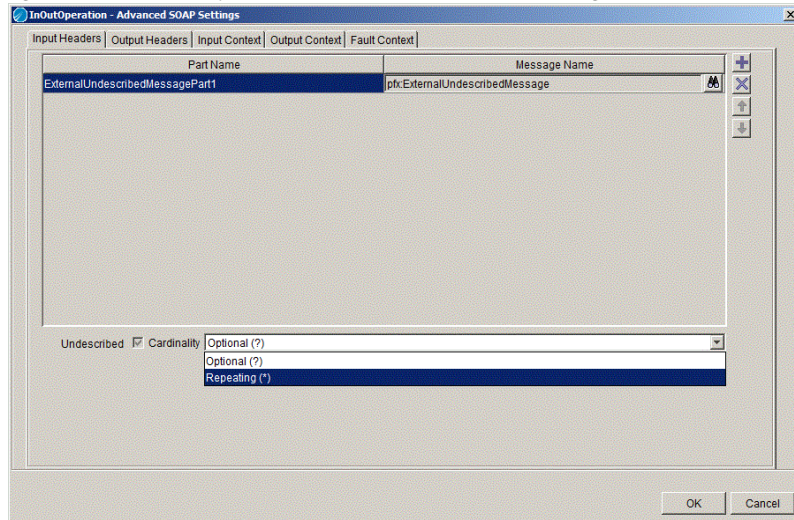
Figure 61 Partner Link Configuration-Advanced



In the selected operations, you can now add the undescribed headers..

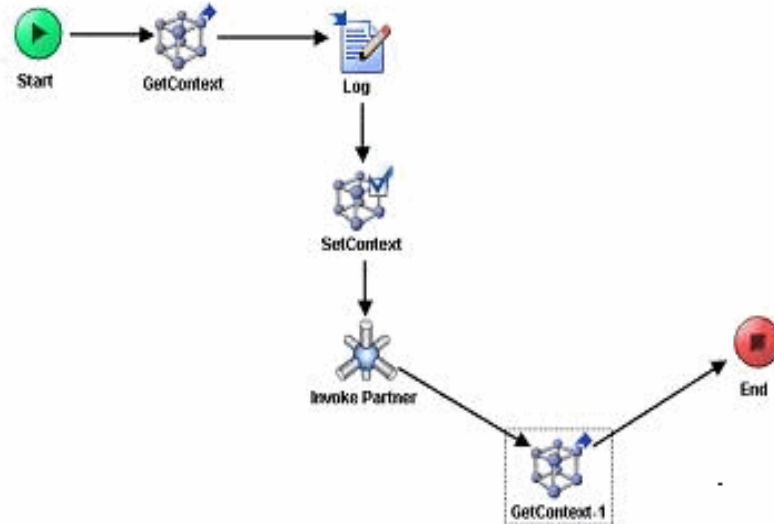
2. In the InOut Operations - Advanced SOAP Settings screen, select the Cardinality; Repeating or Optional and click OK.

Figure 62 InOut Operations- Advanced SOAP Settings



This allows the users to specify headers to be set on an Invocation Request and map context information into these headers.

Figure 63 Activity Sequence



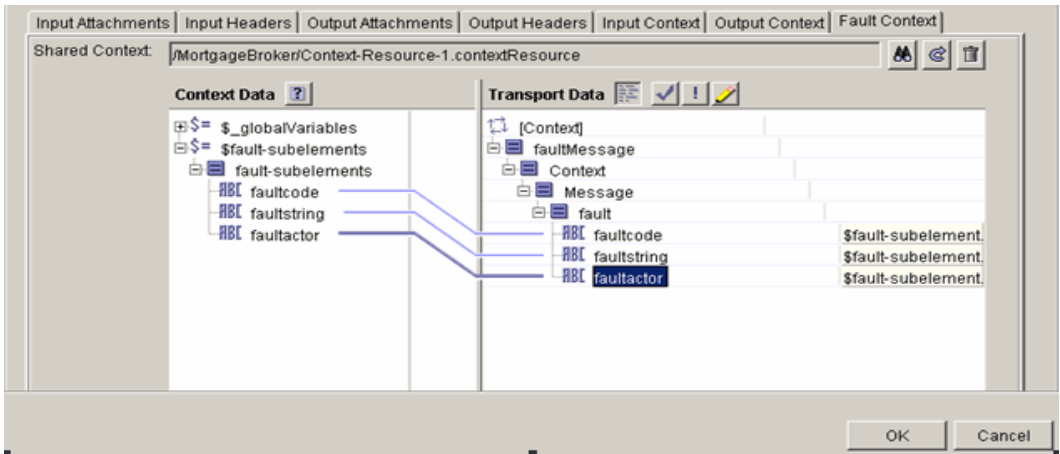
SOAP Fault Sub-Element Propagation

This feature allows the SOAP Fault sub-elements - faultcode, faultstring, and faultactor to be specified on SOAP faults generated by TIBCO ActiveMatrix BusinessWorks. With this mechanism the SOAP faults generated within the TIBCO ActiveMatrix Business Works process can be propagated to the framework in which it is embedded. To configure and propagate the SOAP fault context, the SOAP endpoint configuration user interface for the service and partner link configuration resources are enhanced to include the configuration of fault context.

Fault Context for a Service Resource

Fault Context tab in the **Advanced SOAP Settings** screen allows the user to select a context resource. User can map the fault elements to the context object as shown in [Figure 64](#).

Figure 64 Fault Context Data Mapping in Service Resource

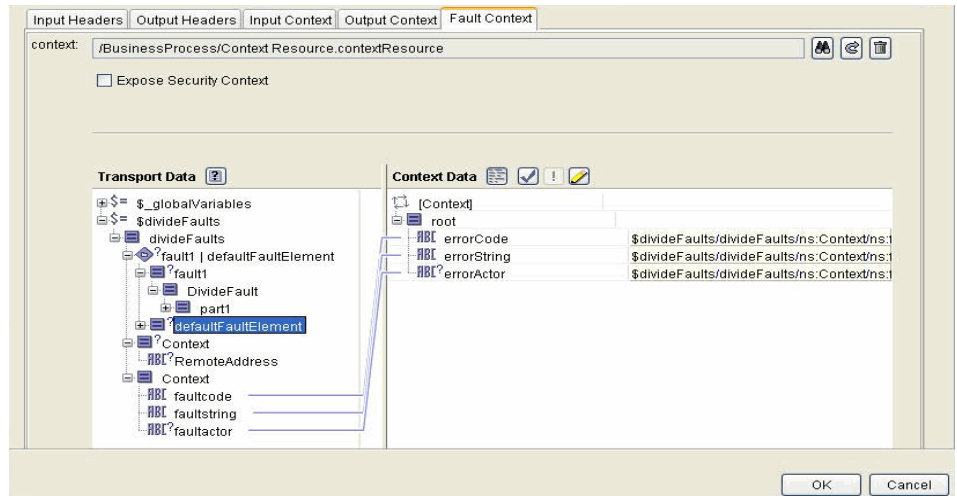


The selection of the shared context resource enables the mapper. The Context Data contains the global variables and the shared context resource data, while the Transport Data contains the basic representation of the context.

Configuring Fault Context on Partner Link Configuration

You can select a shared context resource. The selection of the shared context resource enables the mapper, whose left hand side contains the global variables and the canonical representation of the context, while the right hand side contains the context resource element.

Figure 65 Fault Context Propagation on Partner Link Configuration



Web Service Wizards

Creating web services involves multiple resources and configuration options. TIBCO ActiveMatrix BusinessWorks provides two wizards for easily creating web services. The two wizards are useful in the following situations:

- you wish to expose one or more process definitions as web services.
- you have a WSDL file describing the interface to a web service and you wish to quickly create the process definition stubs that will implement the operations in the interface.

The following sections describe each wizard.

Creating Web Services From Process Definitions

You may have one or more process definitions that you wish to expose as web services. TIBCO ActiveMatrix BusinessWorks provides a wizard that creates a Service resource and WSDL file for the selected process definitions.

To create a service resource and WSDL file from one or more process definitions, perform the following procedure:

1. Open a project in TIBCO Designer and create one or more process definitions that implement operations you wish to expose as web services. The input, output, and error schemas for each process definition must be specified as schema elements defined in the XSD. See [Start Activity on page 67](#) and [End Activity on page 69](#) for more information about defining input, output, and error schemas.
2. Create a transport shared configuration resource (either HTTP Connection or JMS Connection) to use as the transport for the web service.
3. Select and right-click on the name of one of the process definitions in the project tree.
4. Choose **Tools or Multi-User > Generate Web Service > From Process** from the pop-up menu.

The Generate Web Service dialog appears.

- Review the values for the fields in the Generate Web Service dialog. The default values may be sufficient for your purposes, or you may wish to change the values. The following table describes each field.

Field	Description
Namespace	Specifies the namespace to use for the web service.
Port Type	Specifies the name of the interface for the web service. In WSDL terminology, this is referred to as the Port Type.
Transport	Transport for incoming messages. This can be either an HTTP Connection or a JMS Connection shared configuration resource. The default value in this field is the first HTTP Connection or JMS Connection resource found in the project tree.
Location	Location in the project to place the generated Service and WSDL resources.
Operation Names	List of operations in the generated service interface. This list contains the process definition names that implement each operation and the names of the corresponding operations. You can double click on either name to change either the process definition or the operation name. Use the down arrow button to move the selected operation down in the list. You can use the Add More Processes to Interface button to select other process definitions to include in the service interface.
Process Chooser	This field contains the Add More Processes to Interface button. Click this button to view a list of other process definitions you can add to the interface. Select the desired process definitions and click OK to add them to the list in the Operation Names field.

- Click **Generate** to create a Service resource and WSDL file for the selected process definitions.

The Service resource is named `intf<processDefinitionName>-service` and the WSDL file is named `intf<processDefinitionName>`. The service and process definition stubs are placed in the location specified in the Generate Web Service dialog.

7. Click the `intf<processDefinitionName>-service` resource in the project tree to display the service. Ensure the endpoint bindings are specified correctly. For HTTP transports, you may need to specify the Endpoint URI. For JMS transports, you must specify the destination name and other JMS configuration information. See the description of the Service resource in *TIBCO ActiveMatrix BusinessWorks Palette Reference* for more information about specifying endpoint bindings.

Creating Process Definition Stubs From a WSDL File

You may have a WSDL file that describes the services you wish to implement. TIBCO ActiveMatrix BusinessWorks provides a wizard that creates the Service resource and process definition stubs that implement the operations in your WSDL file.

To create a service resource and process definition stubs from a WSDL file, perform the following procedure:

1. Place the WSDL file in your project directory.
2. If your project is not open, open your project in TIBCO Designer. If your project is already open in TIBCO Designer, choose **Resources > Refresh** from the menu.
3. Create a transport shared configuration resource (either HTTP Connection or JMS Connection) to use as the transport for the web service.
4. Select and right-click on the name of the WSDL file in the project tree.
5. Choose **Tools or Multi-User > Generate Web Service > From WSDL** from the pop-up menu.

The WSDL to Process dialog appears.

- Review the values for the fields in the WSDL to Process dialog. The default values may be sufficient for your purposes, or you may wish to change the values. The following table describes each field.

Field	Description
Port Type	Specifies the WSDL file and name of the interface for the web service. In WSDL terminology, this is referred to as the Port Type.
Transport	Transport for incoming messages. This can be either an HTTP Connection or a JMS Connection shared configuration resource. The default value in this field is the first HTTP Connection or JMS Connection resource found in the project tree.
Location	Location in the project to place the generated Service and WSDL resources.

- Click **Generate** to create a Service resource and process definition stubs for the operations in this WSDL file.
 The Service resource is named `<interfaceName>-service` and the process definition stubs are named `<portType><operationName>`. The service and process definition stubs are placed in the location specified in the WSDL to Process dialog.
- Click the `<interfaceName>-service` resource in the project tree to display the service. Ensure the endpoint bindings are specified correctly. For HTTP transports, you may need to specify the Endpoint URI. For JMS transports, you must specify the destination name and other JMS configuration information. See the description of the Service resource in *TIBCO ActiveMatrix BusinessWorks Palette Reference* for more information about specifying endpoint bindings.
- Click on each process definition stub and provide the implementation for each operation by completing the process definition for the operation.

Using Web Services Security Policies

TIBCO ActiveMatrix BusinessWorks allows you to specify security policies for inbound and outbound SOAP messages. The security policies follow Web Services Security: SOAP Message Security 1.0 (WS-Security 2004) OASIS Standard 200401. You can find out more about this standard at <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0>.

You define security policies in the Security Policy shared configuration resource. You can define one policy to share among all of your web services, or you can define multiple policies to use on a per-resource basis. You can include the following attributes within a policy:

- Authentication — whether messages must be authenticated. Authentication can be performed either with usernames and passwords or with X.509 compliant certificates.
- Integrity — whether messages must be validated with a signature to ensure the message has not been altered since its creation.
- Confidentiality — whether messages are encrypted or unencrypted.
- Timeout — whether messages should expire after a certain time.

See *TIBCO ActiveMatrix BusinessWorks Palette Reference* for more information about the Security Policy Resource.



Errors encountered when using Web Service Security policies are generally not published. This is because malicious users could attempt to gain information about your security policy by attempting to replicate known errors.

To prevent the general public from obtaining the Web Service Security error codes, only licensed customers can request a list of error messages and code by contacting TIBCO Support.

Associating Security Policies With Web Services

The Security Policy Association shared configuration resource allows you to associate a security policy with a web service. A security policy can either be associated with individual SOAP resources (SOAP Event Source, SOAP Request Reply, and so on), or it can be associated with each operation in a Service resource. You must create a Security Policy Association resource for each SOAP resource or Service operation to which you wish to apply a security policy.



Security Policy Association shared configuration resources are not referenced by resources in process definitions. Therefore, they are not automatically included in Enterprise Archive files. You must manually add WS Policy Association resources to the Shared Archive within an Enterprise Archive for the associations to work properly in a deployed project. See *TIBCO Administrator User's Guide* for more information about adding resources to the Shared Archive.

When you associate a policy with a Service operation, the policy applies to all regular or fault inbound and outbound messages for the operation. When you associate a policy with a specific SOAP Resource, the appropriate security policies are applied to the messages sent or received by the resource. For example, a SOAP Event Source can only receive messages, therefore a security policy can only be applied to incoming messages. A SOAP Request Reply activity can send and receive messages, and it may also receive a fault message. Therefore, you can associate a security policy for the inbound, outbound, and inbound fault messages.



You can create more than one Security Policy Association resource for the same SOAP or Service resource in your project. This is not recommended because only the first policy association for the resource is used. All other policy associations are ignored.



To run a project with security policy associations successfully, ensure that all the policy associations in the project are valid. Any invalid associations must be removed from the project before running the project.

Custom Password Lookup

By default authentication is checked against the usernames and passwords stored in TIBCO Administrator. TIBCO Administrator can also point to an LDAP registry for username/password lookup. If you store usernames and passwords in your own database or LDAP system, you can write your own password callback class that implements the `javax.security.auth.callback.CallbackHandler` interface.

The CallbackHandler implementation must iterate over each Callback object and look for the WSPasswordCallback type. You can use the WSPasswordCallback.getIdentifyer() method to obtain the username. Then you can write code to lookup the password for that username in your system. Once obtained, you must set the password in the Callback object using the setPassword() method.

Here is a simple example of a CallbackHandler implementation:

```
public class MyPasswordCallback implements CallbackHandler {
    private HashMap passwords = new HashMap();
    public MyPasswordCallback() {
    }
    public void handle(Callback[] callbacks) throws IOException,
        UnsupportedCallbackException {
        for (int i = 0; i < callbacks.length; i++) {
            if (callbacks[i] instanceof WSPasswordCallback) {
                WSPasswordCallback pc =
                    (WSPasswordCallback) callbacks[i];
                String identifier = pc.getIdentifyer();
                int usage = pc.getUsage();
                if (usage == WSPasswordCallback.USERNAME_TOKEN
                    //usage == WSPasswordCallback.USERNAME_TOKEN_UNKNOWN)
                {
                    String pw = (String) passwords.get(sub);
                    if (pw == null) {
                        pw = lookupPassword(identifier);
                        //lookup password using any mechanism.
                    }
                    pc.setPassword(pw);
                }
            }
        }
    }
}
```

For more information about implementing the CallbackHandler interface, see the custom password examples in the BW_HOME/examples/activities/soap directory or see

<http://java.sun.com/j2se/1.4.2/docs/api/javax/security/auth/callback/CallbackHandler.html>.

To use custom password lookup in a security policy, perform the following procedure:

1. Compile your Callback object into a .jar file.
2. Open the TIBCO ActiveMatrix BusinessWorks project containing the processes for which you are creating security policies.
3. Click the General palette and drag and drop an AliasLibrary resource into the design panel.

4. Place your Callback object .jar file into the AliasLibrary resource, and also include any classes that your object depends on (for example, third-party classes for LDAP access). See [Chapter 19, Sharing Common Resources with Other Projects, on page 325](#) for more information about using AliasLibrary resources.
5. Click the Policy palette and drag and drop a Security Policy resource into the design panel. Alternatively, click an existing Security Policy resource to edit the resource.
6. On the Configuration tab, click the checkbox in the Custom Password Lookup field.
7. In the Custom Password Callback Java Class field, use the **Browse** button to locate and select the AliasLibrary resource you created in [step 4](#).
8. In the Class field, use the **Show Class Browser** button to locate and select your Callback object.
9. Click **Apply**, then associate your security policy with the desired web services, if you have not already done so.
10. Save the project.

Security Context Propagation from TIBCO ActiveMatrix Policy Manager

TIBCO ActiveMatrix BusinessWorks allows you to define security policies and associate them with web services such that these policies are applied to the inbound or\and outbound messages by associating security policies with web services. You can now choose not to apply inbound policies in ActiveMatrix BusinessWorks because ActiveMatrix BusinessWorks allows the propagation of security context from TIBCO ActiveMatrix Policy Manager.

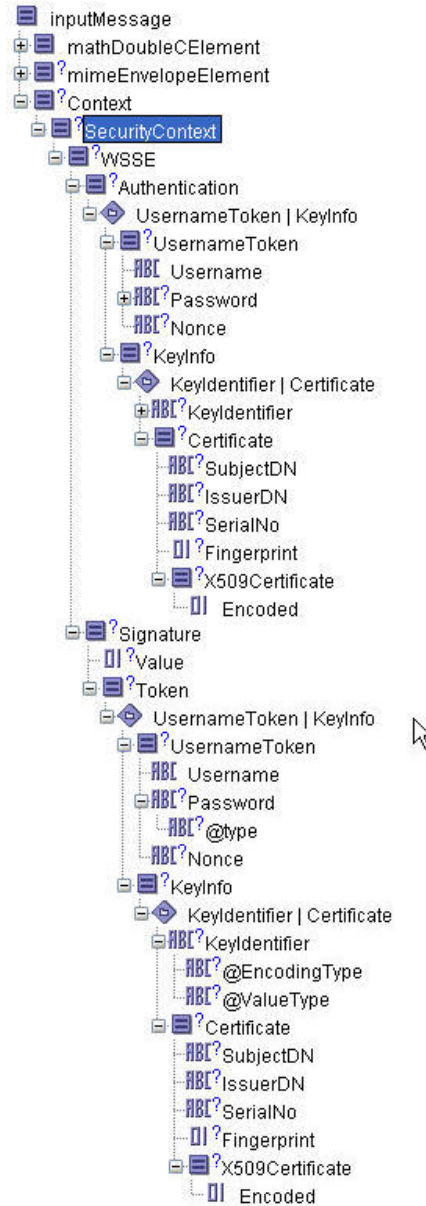
TIBCO ActiveMatrix BusinessWorks can now propagate the security context information sent by TIBCO ActiveMatrix Policy Manager and map it to appropriate fields of the security context in ActiveMatrix BusinessWorks. The security context information is available only when a policy is applied and the information is sent by TIBCO ActiveMatrix Policy Manager.

ActiveMatrix Policy Manager sends security information as an XML document, either as part of the header when HTTP transport is used, or as a message property when JMS transport is used. The security information is available under the header **com.tibco.security.userinformation**.

When ActiveMatrix BusinessWorks Service resource or SOAP Event Source activity receives such a message, it processes the XML document from the header and propagates it in its output tab.

ActiveMatrix BusinessWorks propagates the incoming security context to all outbound invokes by adding the security information as a HTTP header in case of HTTP, or as a JMS message property in case of JMS transport. [Figure 66](#) shows the Authentication Certificate Information and Signature Certificate Information as seen in the Output tab of a SOAP Event Source activity. In this case, the ActiveMatrix Policy Manager had applied the Authentication Signature inbound policy.

Figure 66 Security Context in SOAP Event Source activity Output tab



Enabling security context propagation from TIBCO Policy Manager

You can enable security context propagation from TIBCO Policy Manager by selecting the **Expose Security Context** checkbox available on the Configuration tab of the Service resource or the SOAP Event Source activity using either HTTP or JMS transport.

TIBCO ActiveMatrix BusinessWorks can propagate two types of security context information from ActiveMatrix Policy Manager -

- **Authentication Username Token** - For this policy, TIBCO ActiveMatrix BusinessWorks propagates only the Username value. This is because TIBCO ActiveMatrix Policy Manager does not give out the complete Username Token. Instead, it provides only the Username. The password and the Password type values in the security context will be empty.
- **Authentication Certificate Information and Signature Certificate Information** - For this policy, TIBCO ActiveMatrix BusinessWorks propagates the certificate information, consisting of Subject_DN, Issuer_DN, Serial No and the encoded certificate.

Available Security Contexts in ActiveMatrix BusinessWorks

Table 22 lists the inbound policies available in Policy Manager and the corresponding policies in ActiveMatrix BusinessWorks.

Table 22 Security Contexts Available in ActiveMatrix BusinessWorks

Inbound Policy in ActiveMatrix Policy Manager	Available Security Context in ActiveMatrix BusinessWorks
Authentication - IMS	Authentication Username Token
Crypto - Check Signature	Signature Certificate Information
Authentication Signature and Crypto - Check Signature	Authentication Certificate Information and Signature Certificate Information
Authentication Signature	Authentication Certificate Information and Signature Certificate Information
Authentication - IMS and Crypto - Check Signature	Authentication Username Token and Signature Certificate Information
Authentication - IMS and Crypto - Encrypt and Decrypt	Authentication Username Token and Signature Certificate Information

Limitations to Security Context Propagation from TIBCO ActiveMatrix Policy Manager

The number of security context types that TIBCO ActiveMatrix BusinessWorks propagates is restricted due to certain limitations.

- For Authentication Username Token policy, only the Username value will be available. The password and Password Type values will be empty. This is because ActiveMatrix Policy Manager gives out only the Username in the security context.
- ActiveMatrix BusinessWorks will not propagate any extra information given out by ActiveMatrix Policy Manager. For example, information such as User Role and User Attributes, given by ActiveMatrix Policy Manager will not be propagated.
- ActiveMatrix Policy Manager does not propagate the Transport Security Context received from the client. Hence, the Transport Security Context node in ActiveMatrix BusinessWorks will never be populated with the clients transport security context, when the security context is propagated from ActiveMatrix Policy Manager.
- Signature using Username Token is not available in ActiveMatrix Policy Manager. As a result, it cannot be supported by ActiveMatrix BusinessWorks when using ActiveMatrix Policy Manager.

Using JAAS Login for Authentication

TIBCO ActiveMatrix BusinessWorks uses persistent object framework (POF) API to authenticate with TIBCO Administrator in the standalone mode. Alternatively, you can authenticate TIBCO ActiveMatrix BusinessWorks using JAAS login using the following procedure.

To Authenticate With JAAS Login

1. Run the TIBCO Domain Utility to create an administration domain and enable HTTPS.
2. Create a text file named `jaas.config` and copy the following contents in it.

```
AuthenticationService {
    com.tibco.atlas.auth.jaas.AuthenticationServiceLoginModule
    required
    soap_username="a"
    soap_password="#!1Go16wN7zB+OWxx+eTlB/w=="
    authority="localhost:5443"
    scheme="https";
};
```

Where `soap_username` is the super username of the admin server and `soap_password` is the obfuscated password of the above user.

You can obtain the username and password information from the `AdministrationDomain.properties` file located in `$TIBCO_Home\tra\domain\<domainName>`. The username and password information is provided as a name-value pair with names 'UserID' and 'Credential' respectively. If you find any `\` in the password, you need to remove them as they are escape characters.

`authority="localhost:5443"` specifies admin servers host and https port.



You can create the `jaas.config` file in any folder. However, you need to specify the path of the `jaas.config` file while setting the Java property `com.tibco.bw.security.login.jaas` in `bwengine.xml`.

3. Edit the `bwengine.xml` file and recreate the EAR file before deploying the project.
4. Add the following Java property in `bwengine.xml`:

```
java.property.java.security.auth.login.config=<path>\jaas.conf
```

where `<path>` specifies the path for the `jaas.config` file.

- To use non-default location for keystores (that is, the location other than what JRE uses by default, `jsr\1.5.0\lib\security\cacerts`), add the following properties and set them with appropriate values.

For example, you can use the keystore from admin as follows:

```
java.property.javax.net.ssl.keyStore=C:\Tibco\administrator\domain\<domainName>\SSL\keystore
```

```
java.property.javax.net.ssl.keyStorePassword=password
```

```
java.property.javax.net.ssl.trustStore=C:\Tibco\administrator\domain\<domainName>\SSL\keystore
```

```
java.property.javax.net.ssl.trustStorePassword=password
```

where 'password' specifies the default admin keystore password.

- To use the default keystore, you need to export the certificates from Admin Server's keystore using `keytool` and import them in to jre's keystore.
5. Enable the JAAS authentication by adding the following property in the `bwengine.xml` file:

```
java.property.com.tibco.bw.security.login.jaas=true
```

Following are the JAAS login properties that can be set in `bwengine.xml`. You can refer to these properties while configuring JAAS login:

```

<property>
  <name>JAAS config</name>
  <option>java.property.java.security.auth.login.config</option>
  <default>C:\tibco\jaas\jaas.config</default>
  <description>Used for setting the jaas configuration
  file</description>
</property>
<property>
  <name>Admin Keystore</name>
  <option>java.property.javax.net.ssl.keyStore</option>
  <default></default>
  <description>Sets the Admin Keystore</description>
</property>
<property>
  <name>Keystore Password</name>
  <option>java.property.javax.net.ssl.keyStorePassword</option>
  <default>none</default>
  <description>Sets the keystore password</description>
</property>
<property>
  <name>admin truststore</name>
  <option>java.property.javax.net.ssl.trustStore</option>
  <default>none</default>
  <description>Sets the admin truststore</description>
</property>
<property>
  <name>truststore password</name>
  <option>java.property.javax.net.ssl.trustStorePassword</option>
  <default>none</default>
  <description>sets the truststore password</description>
</property>
<property>
  <name>JAAS Login</name>
  <option>java.property.com.tibco.bw.security.login.jaas</option>
  <default>true</default>
  <description>sets the JAAS to true or false</description>
</property>

```

WSIL Files and UDDI Registries

Universal Description, Discovery, and Integration (UDDI) refers to the protocol used by web-based registries to publish information about web services. Web Service Inspection Language (WSIL) is an XML document format that is similar in function to UDDI but simpler in implementation.

Businesses publish information about the web services they offer using WSIL or to UDDI Operator Sites. This allows other businesses or other groups within a business to locate and access published web services. TIBCO ActiveMatrix BusinessWorks supports both browsing and publishing to UDDI registries that comply with the UDDI Version 2.0 API specification. TIBCO ActiveMatrix BusinessWorks also supports browsing WSIL files that comply with the WSIL 1.0 specification. See

<http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm> for more information about the UDDI API. See <http://www.ibm.com/developerworks/library/specification/ws-wsilspec/> for more information about WSIL.

You can access UDDI registries through TIBCO Designer and TIBCO Administrator. TIBCO Administrator provides a UDDI module that allows you to browse and publish web service details. See *TIBCO ActiveMatrix BusinessWorks Administration* for more information about this module.

In TIBCO Designer, the Tools > UDDI menu brings up the WSIL and UDDI Registry Browser dialog. This dialog allows you to browse web services or publish the details of your web service. The following sections describe the tabs of the WSIL and UDDI Registry Browser dialog.

Connect

The Connect tab specifies the information required to connect to a UDDI server. The following describes the fields on this tab.

Field	Description
Inquiry URL	Select either WSIL or UDDI, then enter the URL for accessing the WSIL files or the UDDI registry. Enter your own URL or use the drop-down list to select one of the available UDDI registries.
Publish Info	
Publish URL	Enter the URL to use for publishing information to the UDDI registry.

Field	Description
User	User name for publishing to the UDDI registry.
Password	Password for the specified user name.
SSL Configuration	
Trusted Root Certs Folder	Folder in the project containing one or more certificates from trusted certificate authorities. This folder is checked when a client connects to the registry server to ensure that the server is trusted. This prevents connections to rogue servers.
Proxy Server Settings	
Specify these fields when you access the registry by way of a proxy server.	
Host	Host name of the proxy server.
Port	Port number on the proxy server.
User Name	User name on the proxy server.
Password	Password on the proxy server.

Browse

The Browse tab specifies the search criteria you wish to use to locate web services. This tab also displays the results of the search. The following describes the fields on this tab.

Field	Description
Search Fields	
Search Type	This field specifies the type of search to perform. Select from the following options: • Service Name • Business Name • Tmodel Key • All WSDLs • NAISC • DUNS • ISO 3166

Field	Description
Search Term	Search term to filter the registry's entries. Only entries with the specified search term are retrieved. The search term is dependent upon the type of search being performed: • Service Name — searches for services that match the specified service name. • Business Name — searches for services that match the specified business name. • Tmodel Key — searches for services that match the specified UUID key for a tmodel. • All WSDLs — searches for services that match the specified service name. Services of tmodel type wsdlSpec are searched. • NAISC — searches for services that match the specified NAICS number. See http://www.naics.com for more information. • DUNS — searches for services that match the nine-digit D-U-N-S number. For more information, see http://www.dnb.com. • ISO 3166 — searches for services that match the specified country code. For a list of ISO 3166 country codes, see http://www.iso.org/iso/en/prods-services/iso3166ma/02iso-3166-code-lists/list-en1.html.
Max Rows	Maximum number of services to return.
Case-sensitive	Checking this box indicates the case of the search term should be considered when locating services.
Order Asc	Checking this box indicates the results should be in ascending order.
Exact Match	Checking this box indicates the search term must be matched exactly.

Field	Description
All WSDLs Area for Searches Based on WSDLs, NAISC, DUNS, or ISO 3166	
WSDLs are listed in the All WSDLs field.	
Get Services Button	Select a WSDL in the All WSDLs field then click the Get Services button to return the service details in the results table.
Results Columns	
Service Name	Name of the service.
Business Key	UUID key for the business.
Description	Description of the service.
WSDL or Access URL	WSDL file or URL for accessing the WSDL file.
Delete Service button	Removes the selected service from the list of returned services.

Publish

The Publish tab allows you to specify web services to publish to a UDDI registry. This tab contains two sub-tabs, Business Details and Service. The following describes the fields on these sub-tabs.

Field	Description
Business Details Sub-Tab	
This sub-tab specifies the details of your company.	
Name	Name of your company.
Key	Business key (UUID) for your company.
Description	Description of your company.
Contact	Name of person or organization to contact in your company.
Address	Address of the person or organization specified in the Contact field.

Field	Description
Phone	Phone of the person or organization specified in the Contact field.
Email	Email of the person or organization specified in the Contact field.

Services Sub-Tab

This sub-tab specifies the web services you wish to publish. Select the web service from the tree on the left and specify the values for the fields on the right.

Name	Name of the web service.
Description	Description of the web service.
WSDL URL	The URL of the WSDL file for the web service.
Endpoint URL	The URL(s) of the deployed web service.

Publish Button

The Publish button publishes the specified web services to a UDDI registry.

Chapter 15 **Using the Schema Import Tool**

The schema import tool allows you to import resources such as XSDs or WSDL files into your project. The tool automatically imports any referenced resources along with the specified resource. This chapter describes how to use the schema import tool.

Topics

- [Overview of the Schema Import Tool, page 282](#)
- [The Location Resource, page 283](#)
- [Importing Resources, page 284](#)

Overview of the Schema Import Tool

The schema import tool allows you to import resources into a TIBCO ActiveMatrix BusinessWorks project. The tool can also resolve references to other resources within the imported resources. The tool can be used to import the following types of resources:

- XSD — XML schema definition language for declaring schema components like elements, attributes, and notations.
- WSDL — Web Services Definition Language for declaring the interface definition of a web services. WSDL files typically include XSD definitions as well.

WSDL and XSD files can contain references to other resources by using import or include statements. The schema import tool can resolve references to other documents and import any referenced documents into the project. The tool helps to avoid such problems as broken references.

To invoke the schema import tool, choose Tools > Schema Importer from the TIBCO Designer menu.

The Location Resource

The Schema Import tool stores imported resources in a Location resource in the project. This resource is a temporary holding area for XSDs and WSDL files while importing and resolving references in the resources. The Location resource corresponds to one XSD or WSDL resource. The Location resource holds the specified resource and any resources referenced by the XSD or WSDL file. You must create a new Location resource for each XSD or WSDL resource you wish to import.

The imported resources remain in the Location resource so that you can resolve any conflicts or errors in the imported resource. When all conflicts and errors have been resolved, the XSD or WSDL resources can be imported into the project.

Once resources have been imported into the project, the Location object stores a pointer to where the imported resources were originally located. You can remove Location resources once the desired resources have been imported into the project.

Importing Resources

To import XSD or WSDL resources into your project, perform the following procedure:

1. Choose **Tools > Schema Importer** from the TIBCO Designer menu or use the Control+Shift+L keyboard shortcut.
2. Click the Down arrow button next to the text Add a New Location at the top left of the Schema Importer dialog. Select New Import Resolver from the popup menu.

A Location resource appears in the project tree in the Schema Importer dialog.

3. In the URI field, enter a valid URI that specifies the location of the resource you wish to import. You can use the Browse button to locate resources in the file system, or you can use the HTTP or FTP protocol to locate remote resources. For example, you can enter `http://www.myCo.com/jswing.wsdl` as a location.
4. In the Download Folder field, specify the folder in the project in which to place the imported resources once any conflicts or errors have been resolved. You can use the Browse button to locate a folder in the current project.
5. Click the **Fetch** button to retrieve the specified resource. The resource and any referenced resources are stored temporarily in the Location resource created in step 2.

You can expand the Location resource to view the retrieved resources. Select a resource and you can view the Information tab or Source tab to see more details about the resource.

6. Any errors are reported on the Errors tab. Select each error and choose the correct option to resolve the error. You may choose to ignore the error, if you wish to import the resource without resolving the error.
7. Click the **Save** button to import the resources into the project. The schema import tool does not allow you to import any resources until all errors have been either resolved or marked to ignore.

Chapter 16 **Working With Secure Sockets Layer (SSL)**

TIBCO ActiveMatrix BusinessWorks can use Secure Sockets Layer (SSL) to provide secure communication. The successor to SSL is Transport Layer Security (TLS), but the term SSL is used synonymously with TLS in this document.

This chapter describes how to use SSL with the activities and resources that support SSL connections.

Topics

- [Overview of SSL, page 286](#)
- [Identity Resources, page 287](#)
- [Trusted Certificates, page 289](#)
- [SSL Configuration, page 293](#)

Overview of SSL

Secure Sockets Layer (SSL) is a protocol that uses public and private keys to secure communication between parties. When an SSL connection is requested, the initiator (or client) and responder (or server) perform a handshake where digital identities, or certificates, are exchanged to ensure that both parties are who each party expects. SSL can also be used to specify an encryption algorithm for the data that is exchanged between the parties.

TIBCO ActiveMatrix BusinessWorks can act as an initiator or a responder in an SSL connection. Several types of connections can optionally use SSL, such as:

- FTP Connection
- HTTP Connection
- JMS Connection
- Rendezvous Transport

In addition, the following activities can also specify SSL connections:

- ActiveEnterprise Adapter activities using JMS or RV transports
- Send HTTP Request
- SOAP Request Reply

The name SSL has been replaced by Transport Layer Security (TLS), but SSL is used as a synonym for TLS in this document. For more general information about SSL, see the TLS specification at <http://www.ietf.org/rfc/rfc2246.txt> or view any of the online tutorials about SSL or TLS on the web.

TIBCO ActiveMatrix BusinessWorks uses digital certificates to validate the identity of parties in an SSL connection. TIBCO ActiveMatrix BusinessWorks requires that both initiators (clients) and responders (servers) must present certificates during the SSL handshake. Typically, only the server is required to present its certificate to the client for verification, but TIBCO ActiveMatrix BusinessWorks enforces a bi-lateral model where both client and server must present certificates.

TIBCO ActiveMatrix BusinessWorks uses the Identity resource to configure the identity of activities that act as initiators (clients) or responders (servers) in an SSL connection. The Identity resource stores the certificate of the activity (initiator or responder) and the location of the folder in the project that contains the trusted certificates of other parties that can participate in an SSL connection.

This chapter describes Identity resources, trusted certificates, and SSL configuration for each activity.

Identity Resources

Identity resources contain identity information that is used to authorize a connection. The responder (or server) in an SSL connection request must have an identity, but the initiator (or client) must also have an identity. The Identity resource can be used to store one of the following types of identities:

- [Username/ Password](#)
- [Certificate/Private Key](#)
- [Identity File](#)

The following sections describe each kind of identity and when each is used.

Username/ Password

This type of identity is used to store a username and password. This is useful when only basic client authentication is needed. This type of identity is not typically used within TIBCO ActiveMatrix BusinessWorks processes.

Certificate/Private Key

Use this type of identity when the public key and the certificate are stored in two separate files. Typically certificates are stored in Privacy-enhanced Electronic Mail (PEM) format. The URL for the certificate and key must be provided, as well as the password for the key.

This type of identity is used when TIBCO ActiveMatrix BusinessWorks acts either as the initiator or responder in an SSL connection.

Identity File

Use this type of identity when the certificate includes the public key information in the certificate file. The URL and file type of the certificate must be provided, as well as the password for the key.

The certificate can be one of the following types of formats:

- Entrust — Entrust file format.
- JCEKS — Java Cryptography Extension Key Store file format.
- JKS — Java Key Store file format.
- PEM — Privacy-enhanced Electronic Mail file format.

- PKCS12 — Public Key Cryptography Standard (12) file format.

Trusted Certificates

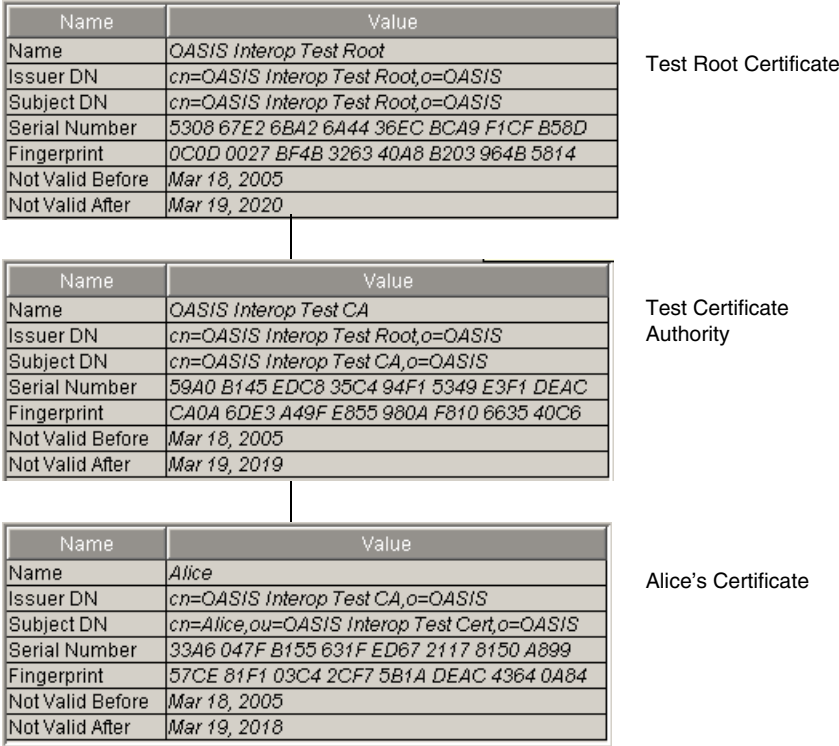
Certificates are typically issued by a trusted third party, such as a certificate authority. There are several commercial certificate authorities, such as Entrust or VeriSign.

Both clients and servers can also store a list of trusted certificates. When a connection is requested, each party presents their certificate and that certificate is checked against the list of trusted certificates. If the certificate is not found, the connection is refused. Checking trusted certificates allows clients to ensure that they are connecting to the correct server. For servers, trusted certificates are used to ensure only the authorized clients can connect to the server.

Checking a certificate involves checking the certificate of the party that signed the certificate. There can be a hierarchy of intermediate certificates, also known as a certificate chain, that must be checked up to the root certificate to ensure that a certificate is authentic. TIBCO ActiveMatrix BusinessWorks requires that all intermediate certificates are stored in the trusted certificate location so that certificates can be properly verified.

[Figure 67](#) illustrates a certificate chain. In this chain, Alice has a certificate signed by the OASIS Interop Test CA (the Issuer DN in the certificate). To verify Alice's identity, the issuer of Alice's certificate must be checked. The OASIS Interop Test CA certificate was issued by OASIS Interop Test Root. You can see that the OASIS Interop Test Root certificate is a self-signed root certificate because the Issuer DN is the same as the Subject DN in the certificate.

Figure 67 An example of a certificate chain



You can use a variety of methods and external tools to obtain certificates for your trusted certificate list or creating self-signed root certificates. For example, OpenSSL at www.openssl.org provides a toolkit for working with certificates and SSL. Portcle at <http://portcle.sourceforge.net/> provides a GUI tool for managing certificates.

TIBCO ActiveMatrix BusinessWorks can import and store certificates in Privacy-enhanced Electronic Mail (PEM) format in a folder within the project. You can also store trusted certificates outside of your project and use a global variable to point to the certificate storage location. The following sections describe each method of storing trusted certificates.

Adding Certificates to Your Project

TIBCO Designer projects store trusted certificates in PEM storage format.

To add a certificate in PEM format to your project, perform the following

procedure:

1. Select a folder into which you wish to import the certificate.
2. From the menu bar, choose **Tools> Trusted Certificates > Import into PEM Format**.
3. Provide the certificate URL when prompted.

You can import certificates that are in PKCS7 and PEM formats (these formats do not store keys). A new certificate copy is created when the import is done. If the certificate to be imported is already in PEM format, a new copy is created as is.

You cannot import certificates from storage formats that require a password, such as PKCS12 and KeyStore.

Storing Trusted Certificates Outside of Your Project

Storing trusted certificates in the project requires you to import any new certificates into the project, re-create the enterprise archive file, and re-deploy your project when certificates change or expire. To avoid this problem, you may wish to store your certificates in a folder outside of your project. When certificates change or expire, you can replace certificates or add new certificates and then restart the process engine to load the changes.

To store trusted certificates outside of your project, perform the following procedure:

1. Create a folder in your file system in the location where you wish to store the trusted certificates. You must copy this folder to each machine where your process engines are deployed, or the location can be a shared network area accessible by all process engines.
2. In your TIBCO ActiveMatrix BusinessWorks project, create a global variable named `BW_GLOBAL_TRUSTED_CA_STORE`. See [Global Variables on page 107](#) for more information about global variables.
3. Set the value of `BW_GLOBAL_TRUSTED_CA_STORE` to the location of the trusted certificates folder on your file system. The location can either be the same for all deployed engines (that is, you copied it to the same location on each machine or it is a shared network drive), or you can change the value of the global variable when you deploy the project to the location on the machine where you place the trusted certificates.

The value you set for `BW_GLOBAL_TRUSTED_CA_STORE` must be a file URL, for example, `file:///c:/tibco/certs`.

4. Specify a value in the Trusted Certificates field in the SSL Configuration dialog. When the project runs, the value of BW_GLOBAL_CA_STORE overrides the value you specify with the location you provided.

SSL Configuration

For connections and activities that allow you to use SSL, there is a checkbox on the configuration that, when checked, allows you to click the Configure SSL button. The Configure SSL button brings up an SSL configuration dialog with specific options for the type of activity or connection that you are configuring. [Table 23](#) describes the potential configuration fields in the SSL configuration dialog for each type of connection.

Table 23 SSL configuration dialog fields

Field	Description
FTP Connection	
FTP Connection resources are used to specify the FTP server that FTP activities will connect to. In this case, TIBCO ActiveMatrix BusinessWorks acts an initiator of SSL connection requests.	
Trusted Certificates Folder	Specifies a folder in the project containing one or more trusted certificates. This folder is checked when an FTP activity connects to ensure that the responder's certificate is from a trusted certificate authority. This prevents connections to rogue servers.
Identity	An Identity resource that contains the client's digital certificate and private key. This field is optional, because clients are typically not required to present their identity to servers.

Table 23 SSL configuration dialog fields

Field	Description
Verify Host Name	Specifies to check that the host name of the FTP 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 Host field on the Configuration tab is not an exact match to the host name specified in the server’s digital certificate, the connection is refused. Note: If you specify an equivalent hostname (for example, an IP address) in the Host field, but the name is not an exact match of the hostname in the host’s digital certificate, the connection is refused.
Strong Cipher Suites Only	When checked, this field specifies that the minimum strength of the cipher suites used can be specified with the <code>bw.plugin.security.strongcipher.minstrength</code> custom engine property. See <i>TIBCO Administrator User’s Guide</i> 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.

Table 23 SSL configuration dialog fields

Field	Description
HTTP Connection	
HTTP connection resources are used when TIBCO ActiveMatrix BusinessWorks acts as an HTTP server. In this case, TIBCO ActiveMatrix BusinessWorks is the responder to SSL connection requests.	
Requires Client Authentication	Checking this field requires initiators to present their digital certificate before connecting to the HTTP server. When this field is checked, the Trusted Certificates Folder becomes enabled so that you can specify a location containing the list of trusted certificates.
Trusted Certificates Folder	This field is only applicable when the Requires Client Authentication field is checked. This field specifies a folder in the project containing one or more certificates from trusted certificate authorities. This folder is checked when a client connects to ensure that the client is trusted. This prevents connections from rogue clients.
Identity	This is an Identity resource that contains the HTTP server's digital certificate and private key.
Strong Cipher Suites Only	When checked, this field specifies that the minimum strength of the cipher suites used can be specified with the <code>bw.plugin.security.strongcipher.minstrength</code> custom engine property. See <i>TIBCO Administrator User's Guide</i> 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.

Table 23 SSL configuration dialog fields

Field	Description
JMS Connection	
TIBCO ActiveMatrix BusinessWorks can act as either an initiator or a responder in an SSL connection with the JMS Connection resource.	
Basic Tab Trusted Certificates Folder	Location of the trusted certificates on this machine. The trusted certificates are a collection of certificates from initiators or responders to whom you will establish connections. If a party presents a certificate that does not match one of your trusted certificates, the connection is refused.
Basic Tab Identity	This is an Identity resource containing the initiator's or responder's certificate. The Browse button allows you to select from a list of appropriately configured Identity resources.
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.
Advanced Tab Debug Trace	Specifies whether SSL debug tracing should be enabled during the connection. Debug tracing provides more detailed messages than standard tracing.

Table 23 SSL configuration dialog fields

Field	Description
Advance Tab Verify Host Name	This field specifies that the host name of the responder should be checked against the host name listed in the responder'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 specified host name is not an exact match to the host name specified in the responder's digital certificate, the connection is refused. If you specify an equivalent hostname (for example, an IP address), but the name is not an exact match of the hostname in the host's digital certificate, the connection is refused. Note: The default context factories for TIBCO Enterprise Message Service automatically determine if host name verification is necessary. If you are using a custom implementation of the context factories, your custom implementation must explicitly set the verify host property to the correct value. For example: <pre>com.tibco.tibjms.TibjmsSSL.setVerifyHost(false)</pre>
Advanced Tab Expected Host Name	This name provided in this field must match the name in the responder's certificate.

Table 23 SSL configuration dialog fields

Field	Description
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 <code>bw.plugin.security.strongcipher.minstrength</code> custom engine property. See <i>TIBCO Administrator User's Guide</i> 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.
Rendezvous Connection	
TIBCO ActiveMatrix BusinessWorks can act as either an initiator or a responder in an SSL connection with the Rendezvous Connection resource.	
Daemon Certificate	Folder containing one or more certificates from trusted certificate authorities. The certificates in this folder are checked when connecting to a daemon to ensure that the connection is to a daemon that is trusted. This prevents connections to rogue TIBCO Rendezvous daemons that attempt to impersonate trusted daemons. You can retrieve a daemon's certificate using the administration interface in TIBCO Rendezvous. See the TIBCO Rendezvous documentation for more information about obtaining certificates through the administration interface. Once retrieved, you can select a folder in your project and import this certificate into the folder using the Tools >Trusted Certificates >Import Into PEM Format menu item.

Table 23 SSL configuration dialog fields

Field	Description
Identity	An Identity resource used to authenticate to the TIBCO Rendezvous daemon. The Browse button allows you to select from a list of appropriately configured Identity resources. Only Identity resources whose Type field is set to Identity File or Username/Password are listed.

Chapter 17 **Creating Custom Activities and Process Starters**

You can package a process definition into a custom activity that hides the implementation details of the process from the user. You can also create a custom process starter either to perform specialized event processing or to handle events not currently supported in TIBCO ActiveMatrix BusinessWorks.

This chapter describes how to create custom activities and process starters.

Topics

- [Overview of Custom Activities, page 302](#)
- [Creating Custom Activities, page 304](#)
- [Using Custom Activities, page 308](#)
- [Custom Process Starters, page 309](#)

Overview of Custom Activities

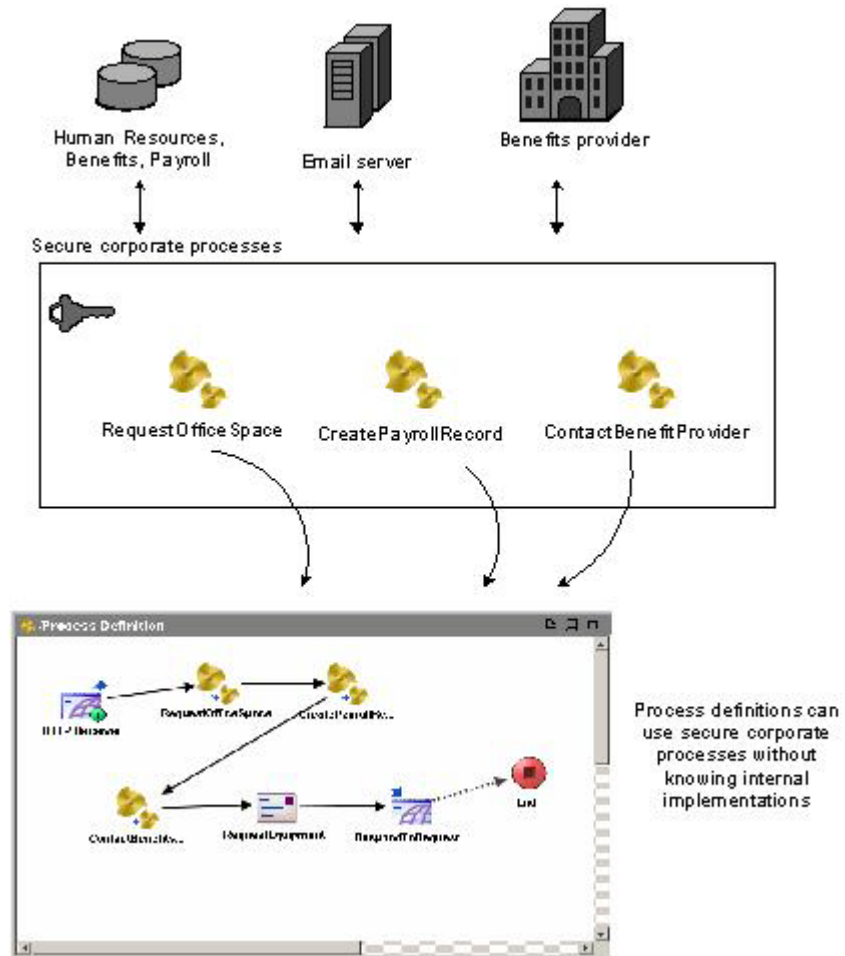
You can package a process definition into a custom activity so that its implementation details are not shown to users. This allows you to create easy-to-use resources that appear as activities in a custom palette, and users can drag and drop these activities into their own process definitions.

For example, your organization may have several internal processes for handling human resources functions, such as setting up office space, creating payroll records, or initiating benefits coverage. For security purposes, the details of how these processes are executed should not be known to the entire company.

You can develop a set of process definitions that automate these business processes and reference them in Custom Activity resources. You can then store the custom activities in a custom palette and distribute that palette to application developers. You could hire outside contractors or use internal resources to develop a larger application such as a corporate human resources portal that uses the custom activities.

[Figure 68](#) illustrates an example of using custom activities.

Figure 68 An example of packaged process definitions



Creating Custom Activities

Before creating custom activities, consider the following:

- Process definitions referenced by a custom activity are referenced by name. You should develop a convention that ensures uniqueness of process definition names, such as storing process definitions in nested folders similar to Java package names (for example, `com/myCompany/Payroll`). This convention helps to avoid naming conflicts with other packaged process definitions that are either created by other groups in your organization or purchased through a third-party.
- The Start activity for a process definition can optionally specify an input schema. This input schema of the referenced process definition is displayed as the input for the custom activity. This schema must contain all information required by the process.
- Only processes that do not have process starters (that is, they begin with the Start activity) can be referenced by a custom activity.
- You may wish to use global variables within process definitions that will be stored in a custom activity. Global variables required by the custom activity are automatically added to the global variable list of the project when the activity is added to a process definition. You should document the required global variables and appropriate values for the variables for users of your custom activities. Also, you should develop a naming scheme for global variables used in your custom activities so that they are unique and do not conflict with other global variables.
- WSDL files or XSD schemas referenced by a process definition are not included when the process definition is placed in a custom activity. You must make WSDL files and XSDs available to any users of your custom activity. These can be imported into a project by the user of the custom activity.

To create a custom activity, perform the following procedure:

1. Drag and drop a folder from the General palette to the design panel. Specify a name for the folder.



It is not required, but it is strongly recommended that you place process definitions into uniquely named folders, or create a folder hierarchy to hold your process definitions. This helps to ensure that the names of process definitions referenced by custom activities will not conflict with each other.

2. Drag and drop a Process Definition resource from the Process palette to the folder you created in Step 1. This process definition defines the business logic you are encapsulating in a custom activity.
3. Optionally, specify an output schema in the Start activity of the process definition. This schema should contain any input you expect users to pass into the custom process.
4. Add activities and transitions to define the business logic for this process.
5. Optionally, specify an input schema for the End activity of the process definition. Map process data into the elements of the input schema. This schema will contain the output data that is available after the custom activity executes.
6. In the project panel, click on the root node of the project, then select the General Activities palette in the palette panel.
7. Drag and drop a Custom Activity resource into the design panel.
8. Specify a name, and optionally specify a custom icon for the custom activity on the Configuration tab of the Custom Activity resource.
9. Click the **Browse** button in the Select Process field and locate the process definition you created in steps 2 through 5 above.
10. Click **Apply**.



Any resources referenced by the process definition in a custom activity are automatically included in the custom activity. For example, your process definition may call other sub-processes or it may reference shared configuration resources. All the referenced resources will be encapsulated in the custom activity along with the specified process definition.

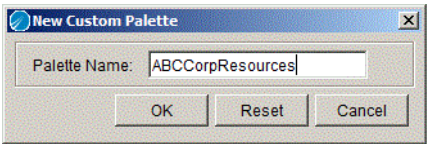
Dynamically called subprocesses are not known at design time, and therefore are not referenced by the process definition. Therefore, dynamically called subprocesses cannot be used by a process definition within a custom activity.

Packaging Custom Activities Into a Custom Palette

TIBCO ActiveMatrix BusinessWorks allows you to create custom palettes that provide a convenient way to package custom activities and other resources for distribution to potential users.

To package a custom activity into a custom palette, perform the following procedure:

1. Create one or more custom palettes to hold the custom activities. To create a custom palette, perform the following procedure:
 - a. Select **Palettes > My Palettes > New Palette** from the TIBCO Designer menu.
 - b. In the dialog that appears, specify a name for the custom palette. For example:



- c. The custom palette appears in the My Palettes tab of the palettes panel.



2. Drag and drop your Custom Activity resources into your custom palette.
3. Save the project.

To distribute a custom palette, perform the following procedure:

1. Locate the file(s) containing the custom palette(s). By default, custom palette files are stored in the location specified in the User Palette Directory field of the General User Preferences dialog. The file extension of the custom palette files is `.mypalette`.
2. Copy the `.mypalette` file(s) to the custom palette directory on the machines you wish to use the custom palette.



You may wish to create an installation program that performs this step so that users can easily install the custom palettes you have created.

3. Start TIBCO Designer and click the **My Palettes** tab to see the custom palette. If TIBCO Designer is already running, choose **Palettes>My Palettes>Reload Palettes** from the menu to retrieve the new custom palette.

Using Custom Activities

Custom activities behave like any other activity in TIBCO ActiveMatrix BusinessWorks.

To use a custom activity, perform the following procedure:

1. Locate the custom palette containing the custom activity on the My Palettes tab.
2. Locate the custom activity within the palette.
3. Drag and drop the custom activity into a process definition in the design panel.



The name of the process definition in the Custom Activity and the name of the process definition in which the Custom Activity is included should be unique. If the names are same, the input and output schemas of the custom activity are not displayed.

4. Create a transition to the custom process and specify the input to the custom process on the Input tab.

Global Variables in Custom Activities

The custom activity may rely on global variables for user-specified information, such as database connection information or JMS topic names. When a user places a custom activity within a process definition, any global variables used by the process referenced in the custom activity that are not already defined in the project are placed in the global variable list of the project.

You should document any global variables required by a custom activity so that users of the activity can specify an appropriate value for each global variable.

Trace Information and Packaged Processes

When tracing is enabled, the trace information for activities executing within a custom activity is written to the log file. This allows you to obtain information that will be helpful if you need to debug a custom activity. When a user of a custom activity experiences a problem, enable tracing and examine the log file to aid in finding the problem.

Custom Process Starters

The Java Event Source process starter allows you to create a custom process starter based on Java code that you create. You may need to create a custom process starter if you wish to do special processing of incoming events before starting a process or if you wish to receive events not currently supported by TIBCO ActiveMatrix BusinessWorks.

The Java Event Source provides the features of other process starters without the need to write extra code. For example, your process starter can take advantage of the Sequencing Key field on the Misc tab without having to write the logic into your Java code.

The Java Event Source resource provides an abstract class that you must extend to create a process starter. The abstract class describes the methods that interact with the process engine. Your code must provide an implementation for the following methods:

- `init()` — this method is called when the process engine starts. This method can initialize any resource connections. Alternatively, you could specify a Java Global Instance on the Advanced tab that initializes resource connections. Java Global Instances are loaded and initialized during process engine start up. You can call `this.getJavaGlobalInstance()` to obtain the Java Global Instance resource in your process starter code.
- `onStart()` — this method is called by the process engine to activate the process starter. This method must activate any event notifier or resource observer code. The notifier or observer code can then call the `onEvent()` method to start a process instance.
- `onStop()` — this method is called by the process engine to deactivate the process starter. This method must deactivate any event notifier or resource observer code.
- `onShutdown()` — this method is called by the process engine when the engine shuts down. This method should release any resources and resource connections and perform any required clean up operations.

The following methods are already implemented and can be used in your code:

- `onEvent(Object object)` — this method is called when a listener or resource observer catches a new event. The input to this method is a Java object containing the event data.
- `getGlobalInstance()` — this method returns an object reference to the Java Global Resource specified on the Advanced tab of the process starter. This is useful if you wish to place initialization code or other shared information in a Java Global Resource instead of in the `init()` method of this class.

- `onError()` — this method throws the exception specified in the input parameter. Use this method to propagate an error to the TIBCO ActiveMatrix BusinessWorks process instance when a listener or resource observer fails to generate an event.

See *TIBCO ActiveMatrix BusinessWorks Palette Reference* for more information about creating and using a Java Event Source process starter.

Chapter 18 **Testing Process Definitions**

This chapter describes the testing mode available for stepping through your process definitions and examining process data.

Topics

- [Overview of Testing, page 312](#)
- [Breakpoints, page 313](#)
- [The Test Panel, page 315](#)
- [Process Instances During Testing, page 316](#)
- [Stepping Through a Process, page 318](#)
- [Colors in Test Mode, page 319](#)
- [Test Mode Buttons and Menus, page 320](#)

Overview of Testing

TIBCO ActiveMatrix BusinessWorks provides a testing environment for stepping through your process models and determining the sources of errors. Entering the testing environment starts a TIBCO ActiveMatrix BusinessWorks engine. The engine starts process instances based on the process definitions stored in your project. You can select one of the running process instances to display in the design panel, and the currently executing activity is highlighted as the process instance runs.

In general, testing should be done during the design and development phase of a project. Testing a deployed project is possible, but might be difficult depending upon the volume of the workload of the system. Also, testing usually involves setting breakpoints in the process model to stop the running process instances at desired points. This is not possible in a production environment, so you may want to use a development system for testing purposes.

Testing a process definition typically involves these steps:

1. Select the process definition you wish to test in the project panel.
2. Set breakpoints in the process definition at points where you wish to stop a running process and examine its state. See [Breakpoints on page 313](#) for more information.
3. If the process begins with a Start activity and the Start activity has a schema defined, you can supply input data to the process before executing it.
4. Click the Tester tab on the left of the project panel. The project panel becomes the test panel. From the test panel you can start process instances or load more process definitions. See [Process Instances During Testing on page 316](#) for more information about process instances in the test panel.
5. Examine the data of the process by selecting any of the activities in the process. The current state of the process data is displayed on the Process Data tab of each activity.
6. Use the toolbar buttons (Pause Testing, Step to Next Activity, and so on) in the test panel to either continue through the process instance or to stop the current process instance. See [Stepping Through a Process on page 318](#) for more information.



Once in testing mode, changes to your process definitions are not reflected in the running process instances. Return to design mode before changing process definitions.

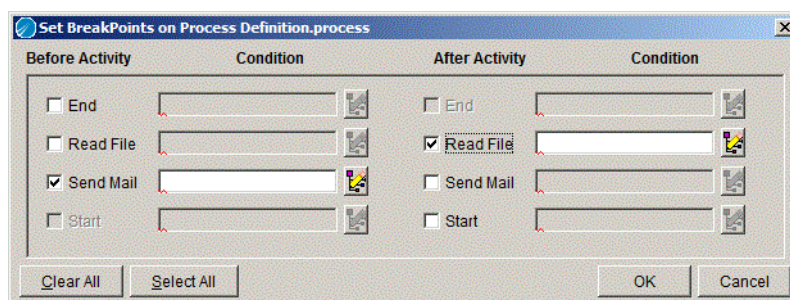
Breakpoints

Breakpoints allow you to suspend a running process instance at a specified point so that you can examine the process data. Breakpoints persist after you close your project — the breakpoints you set should appear in the process definition once the project is reopened.

You can set breakpoints before or after an activity executes. The only exceptions to this are that you cannot set a breakpoint before the starting activity or after the End activity. You can also specify that each breakpoint should only occur based on a given condition. Conditions are specified in XPath, just like conditions for items in an activity's input.

To set a breakpoint, click the Set Breakpoint button  and the Set Breakpoint dialog appears. The dialog allows you to select where to place a breakpoint relative to any of the activities in the current process definition. [Figure 69](#) illustrates an example of the Set Breakpoint dialog.

Figure 69 Set BreakPoints dialog

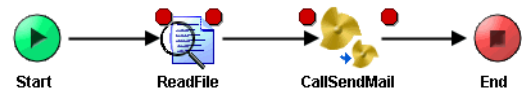


You can choose to select all of the activities by clicking the Select All button. You can clear all set breakpoints by clicking the Clear All button.

You can also set or clear breakpoints on individual activities by right-clicking on the activity and choosing Set/Clear BreakPoint Before/ After from the popup menu. Using the popup menu on the activity only sets the specified breakpoint. You must use the Set Breakpoint dialog if you wish to specify a condition for the breakpoint.

When a breakpoint is set on an activity, a red octagon (a stop sign) appears next to the task's icon to indicate the task has a breakpoint. A breakpoint before the activity appears to the top left of the activity. A breakpoint after the activity appears to the top right of the activity. [Figure 70](#) illustrates a process diagram that has breakpoints set before and after two activities.

Figure 70 Setting a breakpoint



When a process instance is stopped at a breakpoint, the breakpoint icon becomes a stop sign inside a yellow triangle to indicate where the process instance has stopped. [Figure 71](#) illustrates the example process definition when the process instance is stopped at the breakpoint before the ReadFile activity.

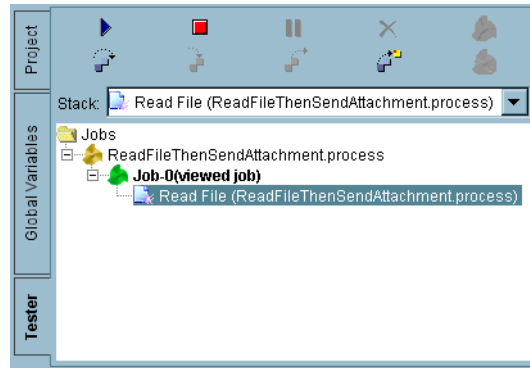
Figure 71 Process instance stopped at a breakpoint



The Test Panel

You can begin testing a process definition by selecting it in the project panel, then clicking the Tester tab to the left of the project panel. The project panel then becomes the test panel. The test panel displays process instances created during testing. [Figure 72](#) illustrates the test panel.

Figure 72 The test panel



The Start Testing Viewed Process button allows you to start process instances for one or more process definitions. You can select process instances in the test panel and display the process definition. See [Process Instances During Testing on page 316](#) for more information about process instances in the test panel.

The test panel has several toolbar buttons for manipulating process instances. See [Test Mode Buttons and Menus on page 320](#) for a complete description of the buttons in the test panel.

Process Instances During Testing

The Start Testing Viewed Process button  allows you to create a process instance for the currently viewed process definition and all of its dependent subprocesses. You can also select other process definitions to load as well.

If the loaded process begins with a Start activity that requires input, you can supply input to the process starter by clicking on the Supply Input Data to Starter button  on the TIBCO Designer toolbar. This button is not available for processes that do not require input data on the Start activity.

Loading Processes to Test

Once the Start Testing Viewed Process button is clicked, the Select Processes to Load dialog appears. You can select the process definitions you wish to load into the test engine in this dialog.

The Advanced button on this dialog allows you to specify any arguments to use when starting the test engine. This is useful if you wish to specify a property file containing custom engine properties. See *TIBCO ActiveMatrix BusinessWorks Administration* for more information about specifying custom properties in a test engine.

You can also specify a database to use for storage of process engine information. The Test Engine Database field specifies the JDBC Connection resource to the database you wish to use.

Creating Process Instances

If the loaded process begins with a Start activity, one process instance is created to execute the process definition. Processing continues until the End activity is reached.

If the loaded process begins with a process starter (for example, Adapter Subscriber), the process engine waits for an incoming event before creating a process instance. Each incoming event causes a process instance to be created, and each process instance is listed in the test panel. You can select any process instance in the test panel and view it in the design panel.

You can create new process instances of any loaded process using the Create a Job button  in the test panel. Select a process definition in the test panel, then click the Create a Job button. A new process instance for that process definition is created.

You can also select and right-click on a process definition name in the test panel to bring up a popup menu. This menu contains the item Create a Job that performs the same function as the button on the test panel toolbar.

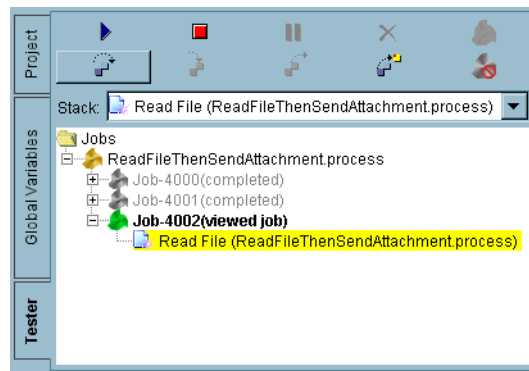
Working With Process Instances

Each process instance is independent in the test panel. You can start/stop/step through each process instance individually.

When a process instance is running, its description is (running) in the test panel. Once a process instance completes its processing (that is, its End activity is reached), its description is changed to (completed job) in the test panel. If a process instance fails to complete, its description is changed to (failed). You can view any running, completed, or failed job.

When a job is selected in the test panel, its process definition is displayed in the design panel and its description changes from (running), (completed job), or (failed) to (viewed job). [Figure 73](#) illustrates process instances and their labels in the test panel.

Figure 73 Multiple process instances in the test panel



You can stop the execution of a process instance by selecting it and clicking the Stop the Current Job button  in the test panel.

You can delete any completed or failed process instances from the test panel by selecting the process instances and clicking the Delete a Completed Job button .



You can browse or change any process definition or any activity's configuration while testing, but changes will not take effect during the current testing session. You must exit test mode and re-enter test mode for changes to take effect.

Stepping Through a Process

When you set a breakpoint in a process definition, the process executes all activities up to the activity with the breakpoint. Once the breakpoint halts processing, you can step through the process using the toolbar icons or menu items. Stepping through the process allows you to examine the executing process instance at your own pace. You can step to the next activity, step into or out of a subprocess, or you can choose another activity later in the process definition and execute from the current point to that later activity. See [Test Mode Buttons and Menus on page 320](#) for more information about the toolbar icons and menu items that allow you to step through a process.

When stepping through a process definition, activities are executed as you pass them. The currently highlighted activity is executed after you choose your next step. If there are multiple paths in a process definition, all transitions that evaluate to true are taken, but only one path is chosen to be highlighted as the next activity when you choose Step to Next Activity.

When you choose to step through a process, breakpoints are still honored, no matter which menu item or toolbar icon you choose. For example, if you are currently in a subprocess and you choose the Step Out of a Subprocess menu item or toolbar icon, execution continues until the next breakpoint occurs or processing of the subprocess completes. If there is a breakpoint before the subprocess completes, processing halts at that breakpoint, and you must choose Step Out of Subprocess again to continue processing.

Colors in Test Mode

When you test a process definition, the elements of the process change colors depending upon what is occurring in the executing process instance. [Table 24](#) describes the colors of each element in a process definition and their significance.

Table 24 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. Therefore the transition has been taken to the next activity.
Red transition arrow	The transition has been evaluated, and its condition evaluates to false. Therefore the transition is not taken.
Red activity	The activity encountered an error while processing. Any Call Process activity that calls a process in which an error occurs is also red.
Bright yellow activity	The process definition 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 the process instance 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 process definition and the focus is not on the current path.

Test Mode Buttons and Menus

There are buttons and icons on the TIBCO Designer toolbar used when testing process definitions. There is also a View > Test Options menu that performs the same actions as the toolbar buttons. [Table 25](#) describes these buttons and menu items.

Table 25 *Toolbar icons for testing*

Button/ Icon	View > Test Options Menu Item	Description
	Set Breakpoints	Brings up the Set Breakpoint dialog that allows you to specify which activities should have breakpoints. Breakpoints stop the process instance and allow you to examine process data before the process continues.
	Add Input Data	Allows you to specify data for the process starter's input schema. This icon is enabled only for process definitions that begin with a Start activity that requires an input schema. This brings up a dialog for creating an input schema. You can use this dialog to save the input data you supply to disk.
		Information icon displayed when a process engine has been started for testing process definitions.
	Go To Started Process	This item is available only on the View>Test menu, there is no toolbar icon. Displays the process definition with which you began this testing session.

Table 25 Toolbar icons for testing

Button/ Icon	View > Test Options Menu Item	Description
	Moving Ball Options	This item is available only on the View>Test menu. There is no toolbar icon. Brings up a dialog that allows you to set whether the moving ball is displayed. The moving ball shows the current execution path. You can also set the speed of the moving ball.

The test panel also has several buttons for manipulating the process instances during testing. The Tools > Tester menu has menu items that perform the equivalent actions as these buttons. [Table 26](#) describes these buttons and menu items.

Table 26 Test panel icons

Button/ Icon	Tools > Tester Menu Item	Description
	Start	Allows you to load the viewed process and select any other processes you wish to load. All dependent subprocesses for any loaded processes are also loaded. The process definitions are loaded into a process engine. Once in testing mode, your process definition cannot be changed. Return to design mode by using the Stop Testing icon if you want to add/remove/change process definitions.
	Resume	Resumes any process instances that are paused or stopped at a breakpoint.

Table 26 Test panel icons

Button/ Icon	Tools > Tester Menu Item	Description
	Stop	Kills the current engine and exits testing mode. All process instances are removed from the test panel. You must click the Start icon to start another engine if you wish to resume testing. Note: In some situations, this button may not stop the process immediately because TIBCO ActiveMatrix BusinessWorks might be waiting for the current operation to be completed. For example, if the current activity is FTP Put and you are attempting to place a very large file on the remote server, the process engine stops only after the FTP command has completed, failed, or a timeout has been reached.
	Pause	Temporarily suspends the process instance. Resume the process instance with the Resume icon.
	Step Over	When a process instance is paused on an activity, click this icon to step ahead in the process definition and execute the next activity.
	Step Into	Once a Call Process activity is reached, this icon allows you to display the process definition of the called process and step through it. This icon is only available when Call Process is the next activity to be processed.
	Step Out	After stepping into a subprocess, this toolbar icon can be used to return to the process that called the subprocess. This icon is only available when you are in a subprocess.

Table 26 Test panel icons

Button/ Icon	Tools > Tester Menu Item	Description
	Show Current Job Location	When a process instance is paused at a breakpoint or any other point, you can change focus to display or edit other resources in your project. This icon and menu item allow you to return focus to the process definition for the currently running process instance. Focus returns to the highlighted activity in a process where the process instance is paused.
		Deletes the selected jobs marked as (completed job) from the test panel. You can only delete completed jobs.
		Creates a new process instance for the selected process definition. You can also select and right click a process definition name in the test panel to bring up a popup menu. This menu has the item Create a Job that performs the same function as the button in the test panel.
	Stop Current Job	Stops the currently executing process instance, but does not exit test mode. This is useful if you wish to examine the data of the process instance, but you do not want to continue running the process.

There are also menu items on the popup menus for each activity in a process definition. You can access these menu items by right clicking on the activity. These are the popup menu items for activities that are used in testing: **Set Breakpoint Before**, **Set Breakpoint After**, **Clear Breakpoint Before**, **Clear Breakpoint After**, and **Run To This Resource**.

The Set/Clear Breakpoint Before/After menu items sets or clears the specified breakpoint on the selected activity.

The Run To This Resource menu item executes the running process instance up to the selected activity. For example, if a process instance is halted on a breakpoint, selecting an activity later in the process definition and choosing the Run To This Resource menu item resumes processing of the process instance and executes all activities between the breakpoint and the selected activity. The process instance pauses when it reaches the activity where you selected the Run To This Resource menu item.

Chapter 19 **Sharing Common Resources with Other Projects**

This chapter explains how you can share resources among projects.

Topics

- [Overview, page 326](#)
- [Creating an AliasLibrary, page 330](#)
- [Creating a LibraryBuilder Resource, page 333](#)
- [Creating a Design-time Library, page 336](#)
- [Managing Resource Conflicts in Design-time Libraries, page 339](#)

Overview

TIBCO Designer allows you to create alias to resources that are to be reused in other projects. Alias are used in two resources, the `AliasLibrary` and the `LibraryBuilder`. The `AliasLibrary` resource allows you to load files stored in the file system into your project. The `LibraryBuilder` resource allows you to build a design-time library that includes resources defined in one project that can be shared with other projects.

AliasLibrary Overview

The `AliasLibrary` resource allows you to specify aliases to file system resources (such as a jar file) that need to be included in your project. To use a file system resource, a project needs to know where to find it. Since projects are exported or deployed to different machines and different environments, Designer uses aliases to specify file locations. Before including a file, an alias is created that specifies the file's location. An alias is part of your preferences and is common to all of your projects. Aliases are created and managed under the `File Alias` tab in the Preferences dialog.

LibraryBuilder Overview

The `LibraryBuilder` resource allows you to share resources you have defined in a project with other project developers. This allows you to create shareable resources once, then allow other project developers to use them in their projects. `LibraryBuilder` resources are used as part of design-time libraries, which are explained later in this chapter.

For example, the following resources can be part of a `LibraryBuilder` resource:

- Schemas (AE or XSD)
- Identities and SSL Certificates
- TIBCO ActiveMatrix BusinessWorks Processes
- TIBCO Rendezvous, JDBC, JMS or other TIBCO ActiveMatrix BusinessWorks shared configurations

A `LibraryBuilder` resource can be maintained using a revision control system or placed in a shared directory. If a `LibraryBuilder` resource is to be shared by many project developers, the resource should be managed in a revision control system where users can sync regularly to get updates. In a smaller environment, `LibraryBuilder` resources can be placed in a shared directory and developers can use email to update each other when there is a change.

If the `LibraryBuilder` resource changes, the resource must be reloaded into your project. The best practice is to minimize changes to the resource. It should be built and tested, then updated infrequently.

To use a `LibraryBuilder` resource, a project needs to know where to find it. Since projects are exported or deployed to different machines and different environments Designer uses aliases to specify libraries locations. Before using a library an alias is created that specifies its location. An alias is part of a user's preferences and is common to all of that user's projects. Aliases are created and managed under the `File Alias` tab in the `Preferences` dialog.

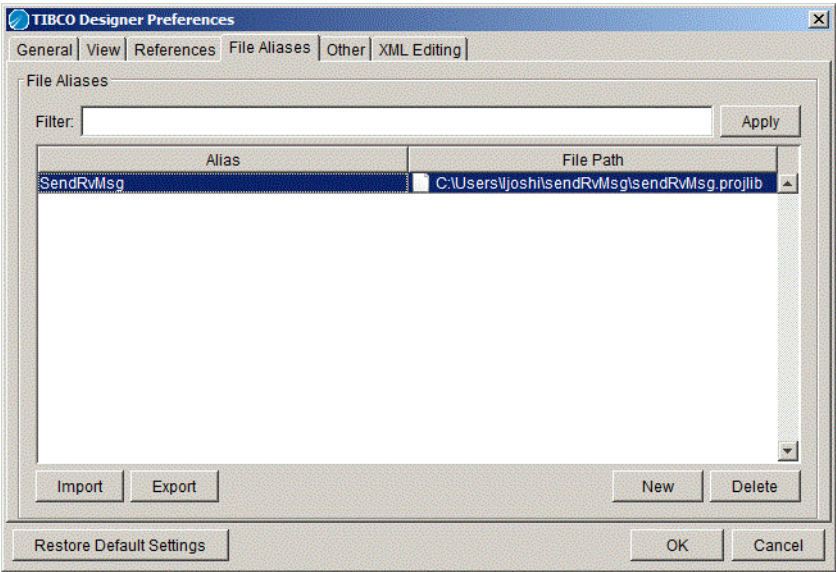
When you build an enterprise archive file from a project that contains a `LibraryBuilder` resource, all resources referenced in the `LibraryBuilder` resource are included in the archive, just as resources in your local project are included.

Creating an Alias

Use these steps to create an alias. After creating an alias, it can be used in an AliasLibrary or LibraryBuilder resource.

1. Click **Edit > Preferences**.
2. Click the **File Aliases** tab.
3. Click **New**.
4. Under the **Alias** column, double-click and provide an alias name.
5. Under the **File Path** column, double-click and navigate to the design-time library location. Select the file name and click **Open**.
6. Click **OK**.

The next diagram shows an alias defined.



Exporting an Alias

1. Click **Edit > Preferences**.
2. Click the **File Aliases** tab.
3. Click **Export** to save the alias list so others can use it.
4. Navigate to a location that others can access.

5. Provide a name for the alias.
6. Click **Save**.

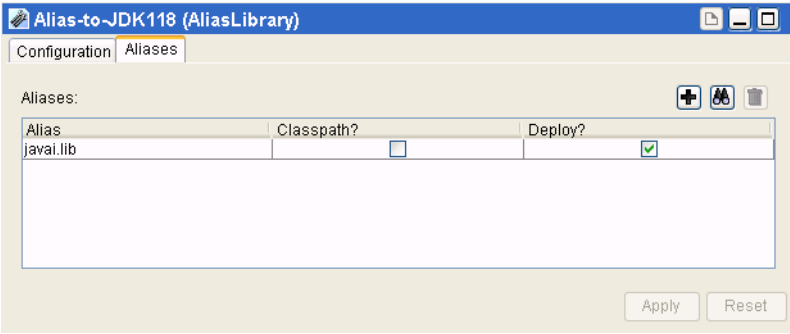
Importing an Alias

1. Click **Edit > Preferences**.
2. Click the **File Aliases** tab.
3. Click **Import** to add a previously exported alias list to your preferences.
4. Navigate to the location of the alias file and select it.
5. Click **Open**.
6. Click **OK**.

Creating an AliasLibrary

The AliasLibrary resource is crucial for resources that depend on external files, such as the Java Activity in TIBCO ActiveMatrix BusinessWorks. Resources in your project can reference aliases in the AliasLibrary to resolve external file dependencies that they may have at runtime or debug time.

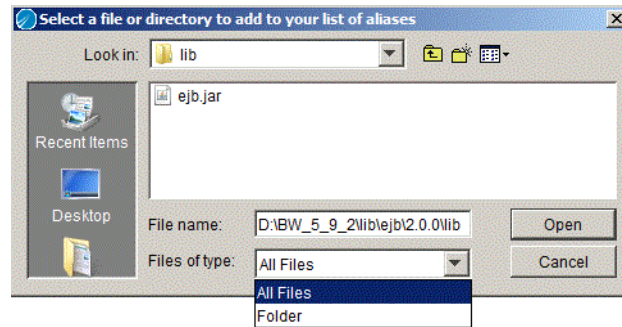
When you build an enterprise archive file, the files referenced by the aliases defined in an AliasLibrary that you include in your project are included in your archive file. The next diagram shows the AliasLibrary resource.



To Create an AliasLibrary

1. Select the top-level folder and locate the AliasLibrary resource.
In palette mode, this resource is in the General palette.
2. Drag the AliasLibrary resource into the design panel.
The AliasLibrary is displayed in the design panel, and the configuration panel now allows you to supply information about the library.
3. Under the **Configuration** tab, provide a name and description for the AliasLibrary.
4. Click the **Aliases** tab.

Click the  icon to select a file or directory to add to your list of aliases. The dialog allows you to pick a Folder or All Files.



Note that if you select a folder, all files in the folder become part of the alias, as well as any sub folders. This means that all files and directories under the folder become part of the enterprise archive file when you build it. If you have a large number of files in the folder, your enterprise archive file will be also be large. When the archive file is deployed, all files in the archive are moved across the network to the remote machines, without regard to which are actually required for the deployment. It is good practice to include only the files required for your project.

A warning appears if the folder of file you select exceeds a certain size. The warning can be customized. See *TIBCO ActiveMatrix BusinessWorks Administration* for more information.

Click the  icon to select a previously defined Alias entry to add to the library. Aliases are managed under the **File Aliases** tab in the TIBCO Designer Preferences dialog. Click **Edit > Preferences** to access the dialog.

The  icon allows you to delete selected entries in the AliasLibrary. Note that the entry is removed from the AliasLibrary. The alias is still available from the preferences dialog.

5. You can specify whether aliases within the AliasLibrary should be included in the CLASSPATH, deployed with the enterprise archive file, or both. This may be required for a TIBCO ActiveMatrix BusinessWorks Java Activity that loads a jar file in order to find its executable or supporting code.
 - If you select the **Classpath** check box, the jar file or file represented by the alias is placed in the CLASSPATH when the enterprise archive file is built. Note that, at design-time, the referring resource is responsible for loading these entries itself. Designer does not perform anything with the configuration other than building the enterprise archive file.
 - If you select the **Deploy** check box, the associated file is deployed. If the **deploy** check box is clear, the file is not deployed. This allows a Java

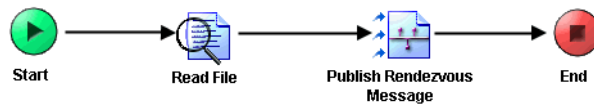
Activity to specify that a support jar file is required in the CLASSPATH, but should not be deployed (the jar may already be available as part of a separate configuration at runtime, but may be required during debug time).

6. Click **Apply**.

Creating a LibraryBuilder Resource

This section explains how to create a LibraryBuilder resource. The next section explains how to include a LibraryBuilder resource as a design-time library and use the resources in the design-time library in a process definition in another project.

The following TIBCO ActiveMatrix BusinessWorks process is used in the example. The process reads a text file and publishes its contents using the TIBCO Rendezvous transport. The design-time library and alias to it is created in the project that contains this example.



Creating a LibraryBuilder resource is a two step process. You first create the LibraryBuilder by naming it, then add resources from your project into it. You then create an alias for the LibraryBuilder so others can use the resource. Aliases are exported so they can be imported into other projects.

Create the LibraryBuilder Resource

1. Select the top-level folder and locate the LibraryBuilder resource.

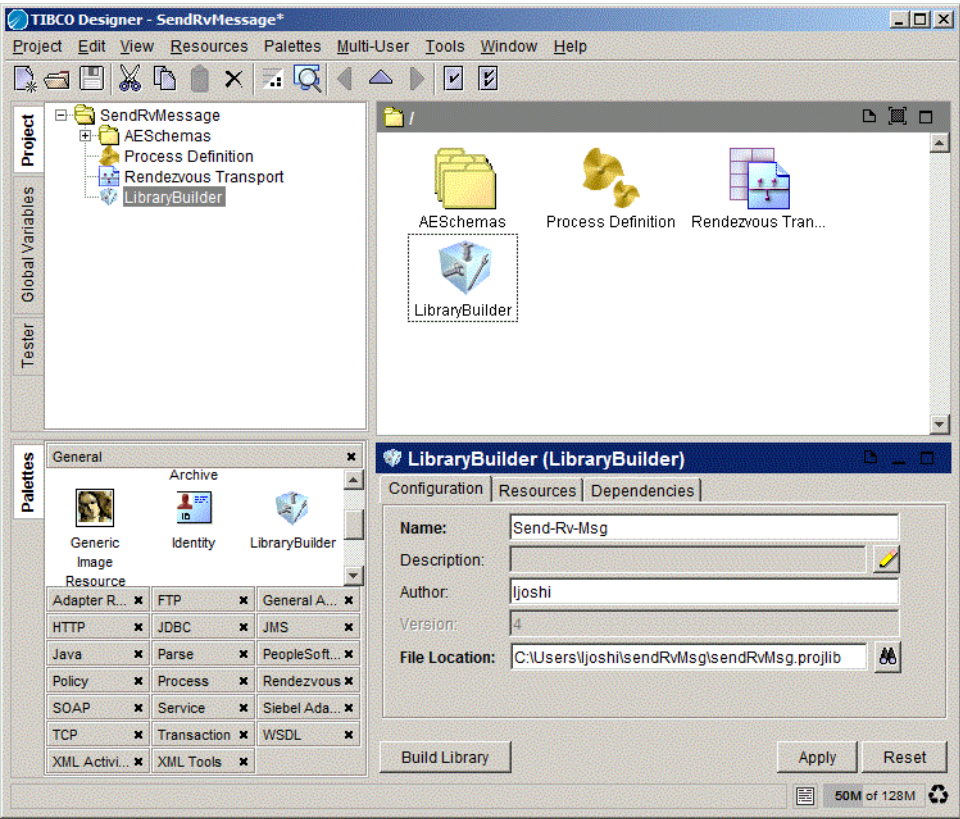
In palette mode, this resource is in the General palette.

2. Drag the LibraryBuilder resource into the design panel.

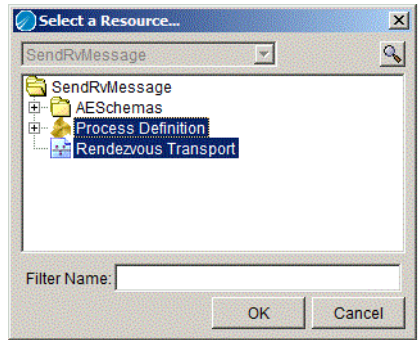
The LibraryBuilder is displayed in the design panel, and the configuration panel now allows you to supply information about the library.

3. Under the **Configuration** tab, provide a name for the LibraryBuilder.

4. In the **File Location** field, provide the absolute path and name for the library file. The directory should be accessible to other developers.



5. Click the **Resources** tab and click the browse icon to select resources to add to the LibraryBuilder. The list can be limited to resources matching a filter term entered in the **Filter Name** field.



6. Click **Apply**. Designer saves the project.

7. Click **Build Library**.

Creating a Design-time Library

A `LibraryBuilder` resource is the basis for a design-time library. Before creating a design-time library, you must have access to a `LibraryBuilder` resource alias.

Aliases are used to manage access to `LibraryBuilder` resources. The aliases you create are saved in your user profile and available to all projects you create on your machine. Other developers can export aliases to `LibraryBuilder` resources, so they can be imported into other projects.

When you add a `LibraryBuilder` resource to a project you can create an alias to it when loading the library, or pick an alias and load the corresponding library.

- If you are working on a different machine, or are not the user who created the alias, you can import an alias to your alias list.
- Clicking **New** when adding a `LibraryBuilder` resource allows you to define an alias and load the `LibraryBuilder` resource in one operation. The alias is then added to your alias list.
- To update a `LibraryBuilder` resource, delete the alias to it, then add it back.
- To delete a `LibraryBuilder` resource from a project, delete the alias to it.

To Create a Design-time Library

1. Open the project to which the `LibraryBuilder` resource will be added.
2. Click **Project > Save**.
3. Select the project's root folder.
4. Click the **Design Time Libraries** tab.
5. Click **Pick** or **New**.

Click **Pick** if you have an alias defined for the library. Select the file alias for the design time library.

Click **New** and navigate to the location of the design-time library. After you click **Apply**, a new alias for the library is added to your alias list.

6. Click **OK**.
7. Click **Apply**.

To Update a Design-time Library

1. Open the project in which the design-time library will be updated.
2. Select the project's root folder.

3. Click the **Design Time Libraries** tab.
4. Select the File Alias for the design-time library to update.
5. Click **Delete**.
6. Click **Apply**.
7. Click **Pick**.
8. Select the file alias for the design time library to update.
9. Click **OK**.
10. Click **Apply**.

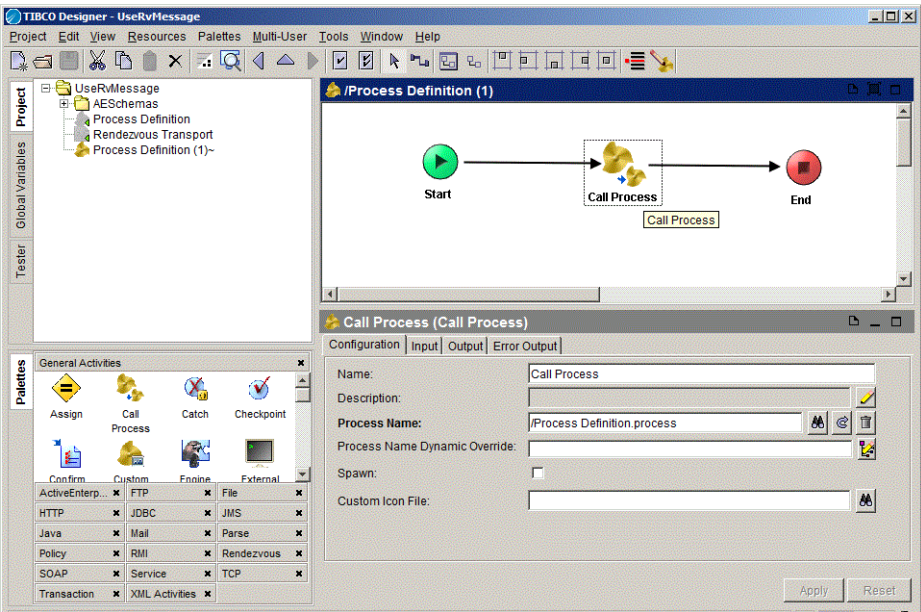
To Delete a Design-time Library

1. Open the project in which the design-time library will be deleted.
2. Select the project's root folder.
3. Click the **Design Time Libraries** tab.
4. Select the File Alias for the design-time library to remove.
5. Click **Delete**.
6. Click **Apply**.

Using a Resource in the Design-time Library

After you have added a `LibraryBuilder` resource to your project, you can use the resources defined in it. The resources display in the project view like other resources but, because they cannot be modified, they are greyed out and not editable.

The next diagram shows a process that calls a process definition in the design-time library. The `Process Name` field points to the process definition in the design-time library.



Showing or Hiding Design-time Libraries

You can hide the design-time libraries that you have added to a project. When you hide the design-time libraries, the resources in the libraries are not available to your project. If you are using a resource in a design-time library and have hidden the library, when you build an enterprise archive file, that resource will not be part of the archive.

To Toggle Showing and Hiding Design-time Libraries

1. Click **Project > Hide Library Resources**.
2. Click **Project > Show Library Resources**.

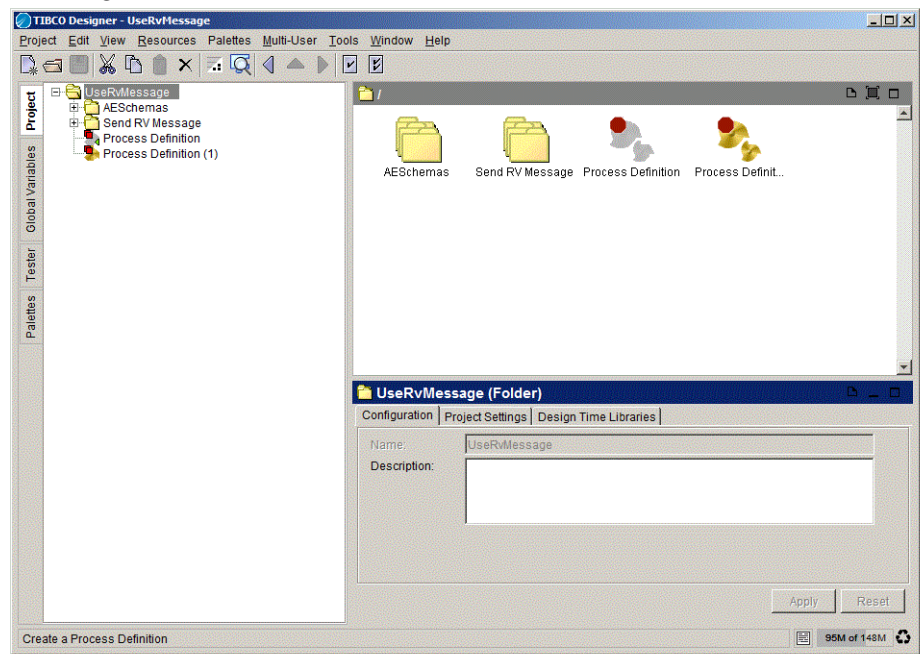
Managing Resource Conflicts in Design-time Libraries

TIBCO Designer prevents you from creating a resource with a name identical to that used by another resource already defined in the project. If a resource in a design-time library has the same name as a pre-existing resource in the project, a conflict arises.

If there is a name conflict, the resource in the local project is always used. If there is a conflict among the design-time libraries you have loaded, you have these choices:

- You can use the arrows on the left to reorder the libraries. Libraries are ordered according to when they were imported. The earliest imported library is at the top of the list. The library at the top of the list has precedence over those lower in the list.
- If reordering does not resolve the conflict, you can re-build the design-time library, or create another one.
- You cannot rename a resource in a design-time library to break a conflict.

Each resource with a conflict is marked so you can easily see them. For example, the following diagram shows a project that contains conflicting resources. You can use the menu command **Resources > View Library Conflicts** to find each conflicting resource.



Appendix A **Working with a Revision Control System**

This appendix gives detailed instructions for working with each supported revision control system. For background information, see the documentation for the revision control system you have chosen.

Topics

- [Overview, page 342](#)
- [File Sharing, page 345](#)
- [Microsoft Visual SourceSafe, page 348](#)
- [Perforce Fast Software Configuration Management System, page 351](#)
- [XML Canon, page 355](#)
- [ClearCase, page 365](#)
- [CVS, page 367](#)
- [PVCS Version Manager, page 370](#)
- [Tips and Tricks for Using Version Control Systems, page 371](#)

Overview

TIBCO Designer allows multiple developers to work on the same project. Developers can use file sharing/locking or a revision control system to ensure that the same resource is not changed by two developers at the same time.

If you wish to use file sharing/locking or a revision control system, you must use a multi-file project. Different users can then add resources to the project and lock the parts of the project they are working on. Note that changes from User A don't show up for User B until User B syncs.



TIBCO Designer creates a file that can be shared and locked for each top-level resource, such as an adapter configuration or a process definition. It does not create a file for each resource. As a result, for example, you can lock an adapter configuration but cannot lock individual adapter services.

TIBCO Designer also creates folders for folders you create in your project. You can lock each folder as needed.

The following revision control options are available:

- File Sharing—Allows you to place the project in a central location, then lock and unlock resources as needed. See [File Sharing on page 345](#).
- Microsoft Visual SourceSafe—Allows multiple users to take advantage of the Visual SourceSafe features. See [Microsoft Visual SourceSafe on page 348](#).
- Perforce—Allows multiple users to take advantage of the Perforce software revision control system. See [Perforce Fast Software Configuration Management System on page 351](#).
- XML Canon—Allows multiple users to take advantage TIBCO XML Canon. See [XML Canon on page 355](#).
- Clearcase— See [ClearCase on page 365](#).
- CVS—See [CVS on page 367](#).
- PVCS—See [PVCS Version Manager on page 370](#).

You interact with the revision control system directly from TIBCO Designer. TIBCO Designer also allows you to check who owns the lock for each locked resource.



Designer may not always have all of the information necessary to distinguish some situations correctly. For example, TIBCO Designer sometimes is unable to distinguish a deleted file from the RCS that should be deleted in your personal copy of the project from a file you added to your copy of the project and want to add to the RCS.

In such cases, use the RCS client directly to fix these situations.

Icons Used by RCS Projects

To illustrate the state of the resource in a project under revision control, TIBCO Designer uses icons on top of each resource in the project panel.



A lock icon indicates that the resource was checked into the revision control system. Other users may be making changes. You need to check out the resource to safely make changes.



A yellow square icon indicates that the RCS does not know about this resource or its state. If the resource is new, you have to add it to the RCS. If it has been checked in before, it has to be checked in again.



A red square indicates, on systems that support that functionality, that another user has locked the resource.

Note that this functionality is not supported for all RC systems.

If no special icon is displayed, the resource has been checked out and is in the same state as the corresponding RCS resource.

Deleting RCS Projects

You delete a project that uses a revision control system as follows:

1. In the Startup panel, click the **Delete project** button (just as for other projects).

2. In the `Delete Project` dialog that appears, supply the information about the project.
 - For projects that use `File Sharing`, use either the `None` or `File Sharing` Revision Control System and any user name, supply the project location.
 - For `Visual SourceSafe` and `Perforce` you must make sure that the project listed in the `Project Directory` field corresponds to the project checked into RCS.
3. Click **OK**.

The project is deleted. For `Visual SourceSafe` and `Perforce`, it is deleted in both the local and the check-in location.

File Sharing

This section discusses using File Sharing as a Revision Control System in the following sections:

- [Preparing for File Sharing on Microsoft Windows on page 345](#)
- [Preparing for File Sharing on UNIX on page 345](#)
- [Using File Sharing on page 345](#)
- [Deleting RCS Projects on page 343](#)

Preparing for File Sharing on Microsoft Windows

The project is located on a shared drive accessible by all TIBCO developers.

Make sure all TIBCO developers have read and write access to that drive.

Preparing for File Sharing on UNIX

The project must be located on a mounted drive accessible by all TIBCO developers. You then go through these steps:

1. Create a Unix group for the TIBCO developers (for example, `tibdev`).
2. Create a Unix account for each developer. Each account must have its Primary Group ID set to the group "tibdev".
3. For each account, the umask must be set to 002 to ensure the entire group has write permission on resources (folder and file) in Designer projects. Set the umask in the `.login` or `.profile` file, as follows:

```
$ umask 002
```

Using File Sharing

Allowing multiple users to use file sharing for a project involves these tasks:

Task A Create the Project

1. Open TIBCO Designer and open the project (which could be a new empty project).
2. Choose **Project > Save > Multi-File Project**.

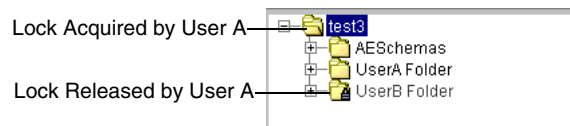
3. In the dialog that appears:
 - a. Supply the root directory for the project (which will become the project name). This directory should be on a drive that can be accessed by all developers that work on the project.
 - b. Choose **File Sharing** from the pop-up.
 - c. Supply your username. Other users will see this username as the owner if you lock files in the project.

Task B User A Acquires Resource and Makes Changes

The creator of the project or another user can now acquire the lock for the project and, for example, add two folders. Here are the steps:

1. User A selects the project root folder.
2. User A chooses **Multi-User > Acquire Resource**.
That command is also available from the resource's right-button menu or from the **Project** menu.
3. User A drags two Folder resources into the design panel and names each for the user who will work with it.
4. User A selects the folder for the second user (User B) and chooses **Release Resource** from the right-button menu.

The folder now appears locked in the TIBCO Designer project panel.



5. User A opens the User A folder and adds two resources to it, then selects each resource and chooses **Add Resource to RCS** from the right-button menu.
User A can make changes to the User A folder (but not to the User B folder).
6. User A saves the project.

Task C User B Opens Project, Acquires Resource, and Makes Changes

A second user can open the same project and make changes to all folders not currently locked by another user. For this example:

1. User B opens the project (all project elements are locked).
2. User B selects the User B folder and chooses **Acquire Resource** from the right-button menu.

User A released the lock for this folder.

3. User B can now make changes to User B folder as desired, and save the project.



User B cannot acquire the lock for the project root folder or for the User A folder because both are locked by User A.

Microsoft Visual SourceSafe

This section first discusses prerequisites and looks at a usage scenario, then includes reference documentation to the Version Control dialog in the following sections:

- [Visual SourceSafe Setup on page 348](#)
- [Using Microsoft Visual SourceSafe on page 348](#)



Microsoft Visual SourceSafe is not supported under UNIX.

Visual SourceSafe Setup

To set up your system to work in conjunction with the TIBCO Designer Visual SourceSafe component, follow these steps:

1. Install a Microsoft Visual SourceSafe 6.0 Client on each machine from which you wish to use TIBCO Designer in conjunction with a Visual SourceSafe database.

Only the Client Programs component is necessary on the machine where TIBCO Designer runs.

2. To make the Visual SourceSafe database available, set the `ssdir` environment variable to the location of the Visual SourceSafe database. The `ReadMess.htm` file included with your client explains how to do this on the command line:

```
set smdir=\\server\share\vss
```

Where `\\server\share\vss` is the folder where the `Srccsafe.ini` file in the VSS database is located.

You can also set this variable permanently using the control panel.



If you do not set this variable, your SourceSafe client cannot find the database where the shared project is located.

Using Microsoft Visual SourceSafe

Step 1: User A Starts TIBCO Designer and Creates a VSS Project

To create a VSS project, User A follows these steps:

1. User A opens TIBCO Designer and chooses New Empty Project.

2. In the dialog that appears, specify the following information:
 - **Project Directory**—Location of the project on the local drive.
 - **Multi-User System**—Visual SourceSafe
 - **User Name**—Name of the current user. The user must have been granted access to VSS during the VSS client installation. Ask your VSS administrator.
 - **Password**—Password for the current user, as specified during the VSS client installation. Ask your VSS administrator.
 - **VSS Command**—Click Browse to point to the SS.EXE executable on your machine. Note that you must use SS.EXE, which is the command-line executable for VSS that is used by TIBCO Designer.
 - **VSS Project Name**—Name of the project in the VSS database. Include the root directory and other directories, as in the example below.

Figure 74 Save Project Options for Microsoft VSS

Save Project

Multi-File Project | XML Canon

Project Directory: C:\tibco\designer15.9\projects\saved

The project directory is the directory that will contain all of the project files. The name of the project will be the name of the project directory. Designer will remove all non-project files from the Project Directory on creation/save.

TIBCO Messaging Encoding: ISO8859-1

Multi-User System: Microsoft Visual SourceSafe **Test Configuration**

User Name:

Password:

VSS Command: <drive>:\<vss_home_dir>\win32\ss.exe

Timeout (seconds): 10

VSS Project Name: \$/<path>/<tibco_project_name>

OK **Cancel**

Step 2: User A Makes Changes to Project and Checks In the Project

User A can now make changes to the project and check them in as follows:

1. Using the TIBCO Designer GUI, User A adds resources to the project and configures them.
2. User A saves the project, then chooses **Multi-User > Add Resource to RCS**.

If a resource has never been added to RCS, you must add. If you make additional changes, you check in the resource.

After User A has supplied a label for this version, the check-in console, which shows the check-in information, is displayed.



All resources are now locked and can be checked out by other users as needed.

Step 3: User B Checks Out Resources and Makes Changes

After User A has checked the whole project into VSS for the first time, each resource can be checked out by each user that has access to VSS.

1. User B opens the project from TIBCO Designer, using the Visual SourceSafe as the Revision Control system and providing a username and password.
2. User B chooses **Multi-User > Project > Synchronize Project** to make sure all resources are loaded.
3. User B selects a resource to be checked out in the project tree, then chooses **Check Out Resource** from the right-button menu of the resource.



If you check out a resource that contains other resources, they may be checked out when you check out the top-level resource.

Whether this happens depends on the directory structure TIBCO Designer creates.

4. User B can now make changes to the checked-out resource. After all changes have been made, User B can choose **Multi-User > Check In Changes**. If User B adds new resource, each resource must first be added to the RCS using the **Add Resource to RCS** menu.



While you can add and checkout recursively, check-in and synchronization is always all or nothing

Perforce Fast Software Configuration Management System

Perforce has comprehensive software configuration management capabilities built around a scalable client/server architecture. Requiring only TCP/IP, developers can access the Perforce Server through a variety of Perforce clients (Windows GUI, Web, or Command-Line). Perforce can be deployed quickly and easily, and requires minimal administration, even for large or distributed sites.

Perforce is supported on a large number of operating systems.

Prerequisites

Before attempting to use Perforce you must ensure the following procedures have been taken.

- Install the Perforce software. TIBCO Designer does not include or install this software.
 - The Perforce server must be installed and running.
 - The Perforce client must be installed on your machine. Only the client is necessary on the machine where TIBCO Designer runs.
- The Perforce server port must be defined.
- Setup a password for your Perforce account. To do this in Microsoft Windows, select **User > Set Password for UserName**. Every Perforce port has a unique password. You may need to define a password for all the Perforce ports you access normally.



Assigning user name and passwords may be done by the Perforce administrator at your site.

- Select the client you use to use or define a new client with any name. You can define a new client in the Perforce **clientSpecs > New** menu.
- Be sure that you have the appropriate permissions to access, create, delete, store, and modify the files you wish to work with under Perforce.

Using Perforce

Step 1: User A Starts TIBCO Designer to Create a Perforce Project

To create a Perforce project, User A follows these steps:

1. User A opens TIBCO Designer and chooses `New Empty Project`.
2. In the dialog that appears, User A specifies the following information:
 - `Project Directory`—Location of the project on the local drive, that is, location where the project is placed by Perforce when you synchronize.
 - `Encoding`—This field is used to determine the wire encoding that TIBCO Rendezvous should use for sending and receiving data in this project. This is a project-wide preference. See *TIBCO Adapter Concepts* for a discussion of how TIBCO adapters support Unicode.
 - `ISO8859-1`—Default value. Use this option if you **ONLY** intend to use TIBCO Rendezvous for English and Western European data.
 - `UTF-8`—Select this option if you want to use TIBCO Rendezvous activities for processing non-Western European data, such as Japanese.
 - `Multi-User System`—Perforce— This selection also enables the **Test Configuration** option. Use this option to test the validity of the information you are using to access Perforce.
 - `User Name`—Name of the current user. This is the same as **Owner** entered in **Perforce Client Specification**.
 - `Password`—Password for the current user, as specified during the Perforce client installation.
 - `Perforce Client`—This is the same as **Client** entered in **Perforce Client Specification**.
 - `Server`—The port on which you access the Perforce server.
 - `Perforce Command`—Click Browse to point to the `P4.EXE` executable on your machine.
 - `Timeout (seconds)`—Amount of time available to connect with the Perforce server before a timeout occurs.



If you set your default client in the Perforce native UI you don't need to fill in the user/server type fields when you access Perforce from Designer

Figure 75 Save Project Options for Perforce Version Control System

Save Project

Multi-File Project | XML Canon

Project Directory: C:\tibco\designer\5.9\projects\saved

The project directory is the directory that will contain all of the project files. The name of the project will be the name of the project directory. Designer will remove all non-project files from the Project Directory on creation/save.

TIBCO Messaging Encoding: ISO8859-1

Multi-User System: Perforce **Test Configuration**

Leave these fields blank, except Perforce Command and Password if required, to use the default values set in the Perforce client.

User Name: my_windows_login_name

Password:

Perforce Client: Perforce_client_name

Server: jupiter.1683

Perforce Command: C:\Program Files\Perforce\p4.exe

Timeout (seconds): 10

Step 2: User A Makes Changes to Project and Checks In the Project

User A can now make changes to the project and check them in as follows:

1. Using the TIBCO Designer GUI, User A adds resources to the project and configures them.
2. User A saves the project, then chooses **Multi-User > Add Resource to RCS**.

If a resource has never been added to RCS, you must add. If you make additional changes, you check in the resource.

After User A has supplied a label for this version, the check-in console, which shows the check-in information, is displayed.



All resources are now locked and can be checked out by other users as needed.

Step 3: User B Checks Out Resources and Makes Changes

After User A has checked the whole project into Perforce for the first time, each resource can be checked out by each user that has access to Perforce.

1. User B opens the project from TIBCO Designer, using the Perforce as the Revision Control system and providing a username and password, and other required information.

2. User B chooses **Multi-User > Project > Synchronize Project** to make sure all resources are loaded.
3. User B selects a resource to be checked out in the project tree, then chooses **Check Out Resource** from the right-button menu of the resource.



If you check out a resource that contains other resources, they may be checked out when you check out the top-level resource.

Whether this happens depends on the directory structure TIBCO Designer creates.

4. User B can now make changes to the checked-out resource. After all changes have been made, User B can choose **Multi-User > Check In Changes**. If User B adds new resource, each resource must first be added to the RCS using the **Add Resource to RCS** menu.



You can add or check in resources recursively.

XML Canon

XML Canon/Developer (XCD) is a comprehensive development platform that allows organizations to store their XML assets (e.g. XML schemas, DTDs, adjuncts, instance documents, and stylesheets) in a central repository that facilitates adaptability, collaboration, and management.



XML Canon is an entire persistence system that has some RCS capability but not a multi-file project.

XML Canon uses permissions to control access to the stored files. XML Canon also provides version control, protecting the development process from duplicate or conflicting efforts.

Features

The following features, accessible via XML Canon's web interface, are provided for the XML-based files in your project, such as your XML Schemas, WSDL files and process definitions.

- Custom Property Association—apply custom metadata to documents or individual components.
- Document and Component-Level Searching—query for document and components through a wide array of filters.
- Namespace Management—browse through a listing of target namespaces and see how a given namespace is used in schema and instance documents.
- XML Document Differencing—track changes between revisions.
- SchemaDOC™—generate a graphical inventory and detailed description of an XML Schema's or DTD's components in a user-friendly HTML format.
- Document/Component Relationship Tracking—track the relationship between documents and their components and determine where schemas or individual components (XML Schema elements or types or WSDL message components, for example) are used within the project.

For more information on XML Canon, see the XML Canon Developer documentation available as online help with the product and also via the TIBCO documentation library. The remainder of this section describes the process for specifying XML Canon as the repository for a project, the typical steps for interacting with XML Canon, and some tips and tricks that will facilitate the effective use of XML Canon.

Prerequisites

To use XML Canon as the version control system for TIBCO Designer, you must have:

- the address of the XML Canon server and the port number on which it is running.
- an XML Canon user name and password with the permissions required to work in the XML Canon category in which the shared project is stored.

Checking In and Acquiring Resources

This section provides the typical steps involved in interacting with the XML Canon repository, beginning with the initial association of the TIBCO Designer project with an XML Canon category.

Step 1: Specifying XML Canon as the version control system for a given project



This step is performed once for a given project. Each project should be associated with a unique XML Canon category.

To specify XML Canon as the version control system for a given project, select **Project > Save As** to display the **Save Project** window. When you open a new empty project, the **Save Project** dialog appears automatically by default.

Select the XML Canon tab, which requires you to specify these fields:

- **Proxy server**—You may access XML Canon via a proxy server that does not require authentication. Not all proxy servers support the WebDAV extensions to HTTP that XML Canon uses. Make sure the proxy server supports required additional functionality.

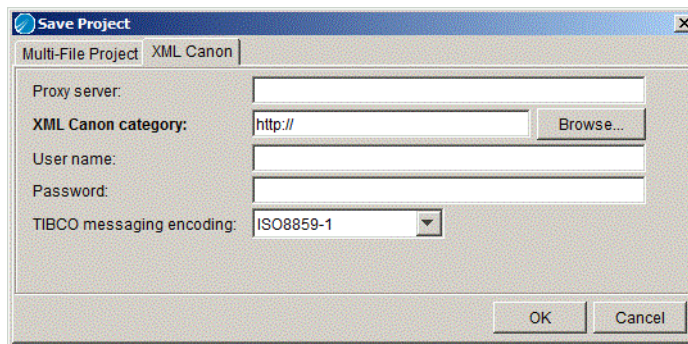


If you click the **Browse** button for the XML Canon category field, the proxy server you specified is taken into account. As a result, it is essential that you specify the proxy server before you click **Browse**.

- **XML Canon category**—the URL (`http://hostName:portNumber/categoryName`) of an empty XML Canon category, which will serve as your top-level project folder.
- **User name**—an XML Canon user name.
- **Password**—password associated with the user name.

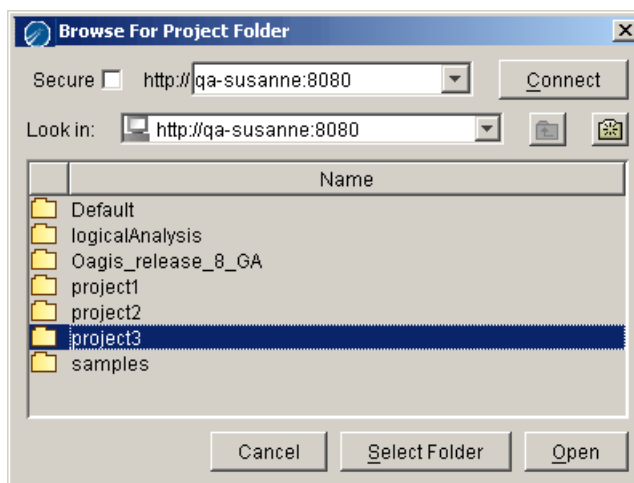
Figure 76 depicts the XML Canon tab filled in with the required information.

Figure 76 XML Canon tab



If you do not know the location of an empty category, click Browse. The Browse for Project Folder will appear. Enter the URL for the XML Canon server and click Connect. You will be prompted to enter your XML Canon user name and password. Upon successful authentication, you will be presented with all of the top-level categories, as depicted in Figure 77.

Figure 77 Browse for an XML Canon category



Double-click a category (or use the Open button) to view its child categories. New categories can be created using the new folder icon. When you have selected the category in which to store the project, click Select Folder.



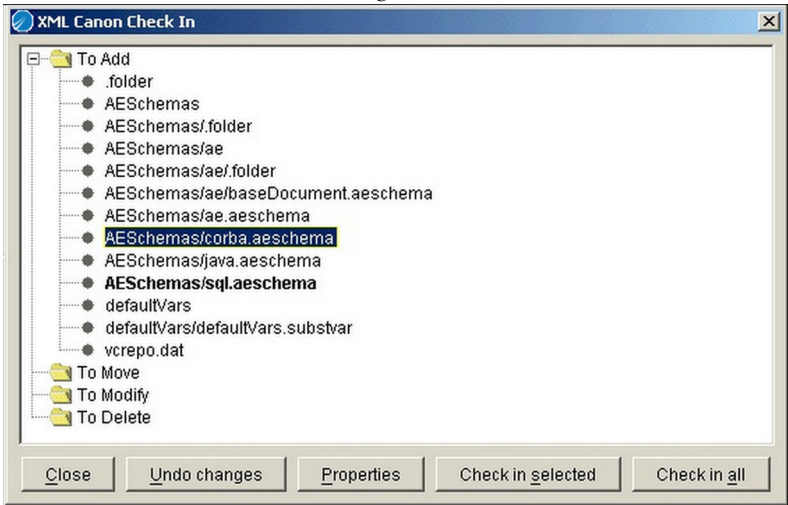
Some XML Canon users may not have the ability to create a new category. If you are unable to create a new category, see your XML Canon administrator.

When a URL, user name, and password have been specified on the XML Canon tab, click OK. The specified XML Canon category will now be used to store the project. See [Step 2: Check in the project](#) to learn how to make the existing resources in the project accessible to all XML Canon users with the permissions to work within the category.

Step 2: Check in the project

The folders and resources existing within the project prior to Step 1 can be checked into XML Canon by way of the Check In Changes option of the Multi-User menu. The XML Canon Check In dialog appears, listing the folders and files that have been added, moved, modified, or deleted since the project was last checked-in. The XML Canon Check In dialog is shown in

Figure 78 XML Canon Check In Dialog



Global variable settings will appear in the dialog as "defaultVars".

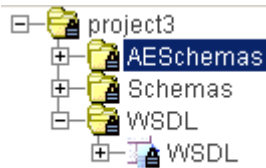
Check-in options are controlled with the following buttons:

Button	Description
Close	Closes the window without checking in any files.
Undo Changes	Reverts the selected folders or resources to their status prior to the last acquire-check out.

Button	Description
Check in selected	Checks in the changes associated with the selected files or folders.
Check in all	Checks in all of the changes.
Properties	XML Canon augments standard WebDAV functionality by allowing you to specify additional properties when a document is saved. By default you may specify a comment, a revision label, and a stage. If the XML Canon Administrator has configured your server to use custom properties, you may also specify those custom properties. To set all properties, you: • Select a file in the "Check In" dialog, then click "Properties...", or • Double-click an item. Items for which the user has already specified custom properties are shown in bold in the dialog. Depending on how the server is configured, it may not be possible to save changes to XML Canon unless you have specified extended properties. This can happen for two reasons. • The "stage" that the document is currently in does not allow a transition to itself (for example, a "production" document cannot be modified without taking it out of "production"), or • The administrator has specified "custom properties" that are required, and the document may not be saved unless those properties are specified.

A lock icon in the project panel, indicating that the files are not acquired and cannot be edited, marks checked-in items. [Figure 79](#) depicts a project checked into XML Canon.

Figure 79 A project checked into XML Canon.



If you are unable to check in changes, your XML Canon user profile may not have the appropriate permission settings. See your XML Canon administrator.

Step 3: Acquiring folders or resources and making changes

Once a project has been associated with XML Canon (see Step 1) and checked-in for the first time (see Step 2), its resources can be acquired (checked out) by other users with access to the XML Canon server and with permission to work within the category associated with the project.

To open a project stored on XML Canon

1. Select the XML Canon tab from the Open Project dialog.
2. Specify (or browse for) the XML Canon category in which the project is stored
3. Supply a user name and password.

Upon successful authentication, the project will be opened.



Upon opening a project shared through XML Canon, you should synchronize the project (Multi-User> Synchronize Project). Synchronize often to ensure that your project tree reflects any resources you have added outside of TIBCO Designer (through a WebDAV folder or the XML Canon interface, for example) as well as any changes made by other users.

Folders and resources marked by a lock icon are read-only until acquired.

To acquire a resource

1. Select the resource in the project tree
2. Select **Acquire-Check out Resource** from the right button menu or the Multi-User menu.

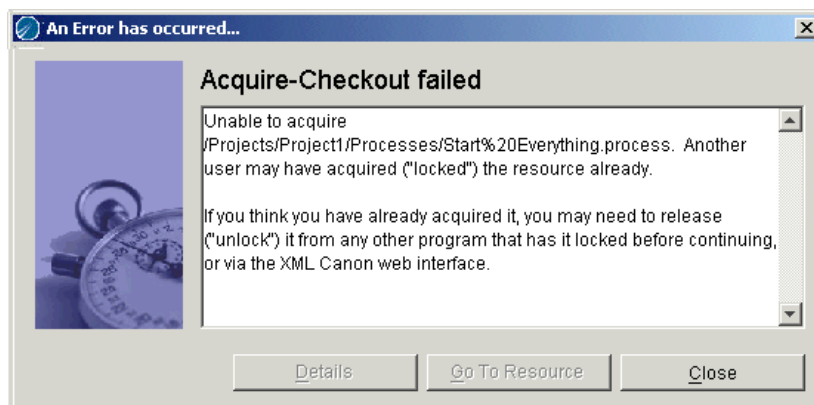
When a resource is acquired, its lock icon will disappear, indicating that you may edit the file.



Use the global variables display to acquire global variables. Global variables are acquired through the Acquire-Check out Global Variable Group option of the right-button menu.

Once a user acquires a resource, other users cannot modify it. (Other users can view the resource, but cannot make changes to it.) If you attempt to acquire a resource that is under the control of another user, the message shown in [Figure 80](#) will appear.

Figure 80 Acquire failed



Step 4: Checking in (or reverting) the changes made to an acquired resource

Changes made to acquired resources can be checked into XML Canon, following the same procedure outlined in Step 2.

Reverting Changes

There are two options for returning a folder or resource to its status prior to your acquisition:

- Select the change(s) within the Check In Changes Dialog and click Undo Changes.
- Select the modified folder or resource within the project panel and select the Release-Revert Resource option, available via the right-button menu and the Multi-User menu.

Viewing Revision Control Information

The revision control system (RCS) information for a particular resource can be viewed by selecting the View RCS Info for Resource option from the right-button menu or the Multi-User menu. The information available is described in [Table 27](#).

Table 27 Revision control system information

Field	Description
Analyzed state	This indicates if the resource was analyzed for document and component level relationships. This applies to XML Schema, DTD, and WSDL resources only.
Comment	Displays any comments added when checking the resource into XML Canon.
Creation date	The date the resource was created.
Display name	The name of the resource.
Document flavor	The resource type.
Document id	The XML Canon internal id for the resource.
Document size	The size of the resource in bytes.
Governing namespace	Namespace that the document governs, for example, in XML Schema, the value of the 'targetNamespace' attribute.
Last modified	The time and date of the last modification.
Last modified by	The name of the user making the last modification.
Last modified by id	The XML Canon internal id of the user making the last modification.
Lock token	Globally unique identifier for the resource being acquired (locked on the server).
Locked by	The user who has acquired the resource.
Mime type	The Multipurpose Internet Mail Extension.
Resource type	The WebDAV resource type.

Table 27 Revision control system information

Field	Description
Revision name	A revision label added when checking the resource into XML Canon.
Revision number	The number of times the resource has been checked into XML Canon.
Root namespace	The default namespace of the resource.
Stage id	The XML Canon internal stage id for the resource.
Stage name	The XML Canon stage of the resource.
Supported lock	The WebDAV locks allowed on the resource.



You can apply custom metadata to the resources through the DAV tab of the user's XML Canon home page. For XML Schemas, DTDs and WSDL files, custom properties can be specified for individual components as well.

Deleting XML Canon Projects

XML Canon-based projects cannot be deleted in TIBCO Designer. To delete a project, use a WebDAV client.

Tips and Tricks

The following tips will help you use XML Canon effectively as a version control system.

- Synchronize (**Multi-User > Synchronize Project**) often to ensure your project reflects the changes made by other users.
- Any conflicts discovered during the synchronization process will be listed along with instructions for resolving the conflict. You cannot check in any changes until synchronization conflicts are resolved.
- Because XML Canon automatically treats all of the project's resources as part of the Revision Control System, the Add Resource to RCS option of the right-button menu Multi-User menus is not applicable when working with XML Canon.
- When checking in changes, keep in mind that some changes are dependent upon other changes. (For example, a new child folder cannot be checked in until its parent folder is checked in.) If you check in a single change (using the

Check in selected button) with a dependency on another change, the other change will also be checked in.

- Acquisitions are not recursive. When you acquire a folder, its resources and sub-folders are not checked-out.
- A folder must be acquired if you want to:
 - a. rename the folder (or one of its resources or sub-folders)
 - b. move the folder
 - c. edit the folder's description
 - d. delete the folder

ClearCase

ClearCase tracks the file versions used to build a product release and helps to organize the work of groups of engineers.

ClearCase requires that a directory element be checked out before any elements within it can be checked out. This implies:

- The root of a newly created project must be saved in a directory that has already been checked out in ClearCase. Otherwise the root (and hence the project) may not be added to the RCS.
- Checking out a non-folder resource will auto-checkout the parent folder resource, if not already checked out in the RCS.

Creating or Modifying a ClearCase Project

To Create a ClearCase Project

1. Start TIBCO Designer.
2. Select **New empty project**.
3. In the **Save Project** dialog, select **ClearCase** from the **Multi-User System** combo box.
4. In **Project Directory**, browse to your ClearCase View and provide a project name. For example, `Z:\<view>\<projectname>`.
5. Provide your ClearCase user name.
6. In **Cleartool Command**, browse to your ClearCase installation and select the ClearCase executable, `cleartool.exe`.
7. Select the **View Type**.
8. Click the **Test Configuration** button, then click **OK** if the test configuration is successful.
9. In the **Project** window, select the project root folder, right click and select **Add Resource to RCS**.
10. Select **Multi-User>Check in Changes** to check in the project into ClearCase.

To Modify a ClearCase Project

1. Open the project

2. Select the project root folder and right click and select **Acquire-Check out Resource**.
3. After modifying the project, save it, then click **Multi-User>Check In Changes**.

CVS

CVS is a version control system that records the history of source files used to build a software product. CVS also allows engineers to work in their own directory, and merges each engineer's work into a source tree.

The following limitations apply when using CVS as an RCS system for TIBCO Designer 5.x:

- By default, CVS does not lock files. All CVS checkouts must be performed with the CVS `-r` option. For example:

cvsv -r checkout -P myproject

This enables RCS to use watches and enables the RCS adapter to limit editing of resources to a single user.

- CVS does not handle binary files correctly unless it has been correctly configured. Before placing a resource under CVS control, verify that the resource's data is *not* in binary format. If it is, configure CVS to handle the appropriate resource `vfile` extension as binary. If you do not do this, corrupted data will result.
- The CVS RCS adapter has only been tested in server mode using the CVSNT available from <http://www.cvsnt.com/cvspro/>. Other servers may work, but have not been tested.
- The CVS RCS adapter has only been tested with `local`, `pserver` and `sspi` protocols. Other protocols should work, but have not been tested.
- CVS requires that files be deleted from the file system and committed before they can be scheduled for deletion. Therefore TIBCO Designer *forgets* about the resource, and cannot restore it for you on a revert operation. To restore a deleted file, use your CVS tools as directed by your CVS documentation.

Working with CVS

The following steps assume you are working on a UNIX platform. If you are working on a Windows platforms, the steps are the same, but the examples apply only to UNIX.

Installing CVS on UNIX

1. Download the installer from <http://www.cvsnt.com/cvspro/?lang=EN&complete=1>.

2. Set PATH to execute CVS commands on your shell. For example, assuming CVS is installed under /usr/local/bin:

```
setenv PATH /usr/local/bin:$PATH
```

3. Set CVSRROOT:

```
setenv CVSRROOT /home/corp/<user>/rcs/cvs
```

4. Initialize CVS. This creates the CVS repository under CVSRROOT:

```
cvs init
```

Importing a Project Into a CVS Repository

1. In TIBCO Designer, create a project.
2. Save the project. For example: /home/corp/<user>/cvstest01
3. Close TIBCO Designer.

4. Using a shell, navigate to the project directory. For example:

```
cd /home/corp/<user>/cvstest01
```

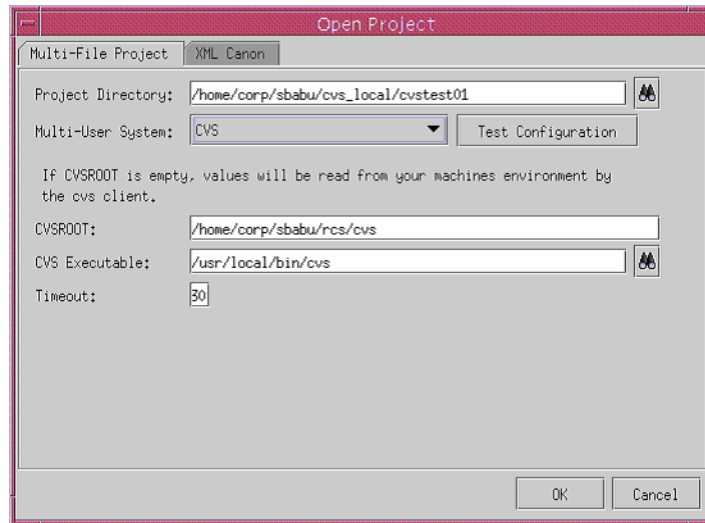
5. Execute the following commands:

```
cvs import -m "comment" <project name> vendor release
cvs import -m "cvstest for designer" cvstest01 vendor release
```

6. Check-out the project to local directory.
7. Using a shell, navigate into the CVS local copy directory. For example:

```
/home/corp/<user>/cvs_local
cd /home/corp/<user>/cvs_local
cvs -r checkout -P <project name>
cvs -r checkout -P cvstest01
```

8. Now you are ready to open the CVS project in TIBCO Designer.



PVCS Version Manager

PVCS allows developers to manage software development by controlling access and revisions to source code. It automates and simplifies access to development objects including source code, ASCII files, graphics, documentation, and binary data files, allowing developers to share archive information and manage projects, even in cross-platform development environments.

Creating or Modifying a PVCS Project

To Create a PVCS Project

1. Start TIBCO Designer.
2. Select **New empty project**.
3. In the Save Project dialog, select **PVCS** from the **Multi-User System** combo box.
4. Provide your PVCS user name and optionally provide a password.
5. Provide the path to your PVCS project name, project database, executable. For example, for the project database:

`C:\Program Files\Merant\vm\common\<Database Name>`

6. Click the **Test Configuration** button, then click **OK** if the test configuration is successful.
7. In the Project window, select the project root folder, right click and select **Add Resource to RCS**.

To Modify a PVCS Project

1. Open the project
2. Select the project root folder and right click and select **Acquire-Check out Resource**.
3. After modifying the project, save it, then click **Multi-User>Check In Changes**.

Tips and Tricks for Using Version Control Systems

The following techniques will help you use your version control system effectively:

- Check in and synchronize on a regular basis. The information displayed by TIBCO Designer may not be completely accurate if there are a large number of differences between the project in TIBCO Designer and the project in VSS.
 - Structure your project so each user owns a folder and works in it. When several users work in the same folder, only the folder owner can add, delete, and rename resources in that folder.
 - Structure your project so that each user owns a folder in the AESchemas area.
4. Do not keep the root folder locked. If you do, other users cannot add resources in it (not even their own folder). Note that this tip does not apply if you are using File Sharing. Even if User A locks the root folder or any other folder, User B is still able to add a folder or resource under a locked folder (root or else).
- Do not keep the AESchemas folder locked. If you do, other users cannot add resources in it (not even their own folder inside AESchemas).
 - Sometimes a check-out is recursive (optional or forced), and sometimes you can add resources to folders even if someone else has checked out that folder.



Do not lock folders unless you have to. Even though the capability of adding to a locked folder is there, it can still cause problems, for example, if two people attempt to add a resource with the same name.

Index

Symbols

`$_error` process variable 176

A

activities 39, 48
 creating groups 78
 End 69
 Input tab icons 121
 Start 67
 stepping through when testing 318
 ungrouping 79
 Adding
 Global Variable or Global Variable Group 108
 adding adapter resources, adapter resources
 adding 29
 addressing schema elements 162
 AliasLibrary Overview 326

B

breakpoints 313
 icon when set 314
 business processes 34

C

Call Process activity 43
 category mode 15
 changing current selection 11
 ClearCase 365
 clearing breakpoints 313

colors used when testing process definitions 319
 conditions 75
 constants in XPath 126
 Creating
 a ClearCase Project 365
 a Design-time Library 336
 a PVCS Project 370
 Alias 328
 AliasLibrary 330
 an AliasLibrary 330
 ClearCase Project 365
 Design-time Library 336
 LibraryBuilder Resource 333, 333
 PVCS Project 370
 creating
 groups 78
 transitions 74
 creating custom palettes 17
 creating projects 24
 current selection 11
 custom palettes 17
 Custom Password Lookup 265
 CVS 367

D

data
 sent across process instances 218, 220
 database storage
 inter-process communication 223
 debugging 311, 312
 breakpoints 313
 icons 320
 Deleting
 a Design-time Library 337
 a User Palette 18
 Global Variable 109

E

- edit custom palette [17](#)
- End activity [69](#)
 - process error schemas [181](#)
- enforcing order of execution [224](#)
- enterprise computing [34](#)
- error handling [174](#)
 - error propagation [179](#)
 - no-action groups [88](#)
 - overview [175](#)
 - process error schemas [181](#)
 - repeat on error until true group [101](#)
- errors [120, 120](#)
- errors in mappings [120](#)
- evaluation context [163](#)
- examples
 - inter-process communication [224](#)
- Exporting
 - Alias [328](#)

G

- Generate Error activity
 - error propagation [179](#)
 - process error schemas [181](#)
- Global Variables
 - Attributes [109](#)
- groups [41, 78](#)
 - creating [78](#)
 - error propagation [179](#)
 - loops [93](#)
 - accumulate output [93](#)
 - index variable [93](#)
 - no action [88](#)
 - ungrouping [79](#)

H

- handling errors [174](#)
- Hiding Design-time Libraries [338, 338](#)

- hints [120](#)

I

- icons
 - test mode [320](#)
- Importing
 - a Project Into a CVS Repository [368](#)
 - an Alias [329](#)
- incoming events
 - key [54](#)
 - scalability [226](#)
 - timeout [54](#)
- index variable [93](#)
- Input tab
 - icons [121](#)
- Insert Model Group Content [132, 132, 132](#)
- Installing CVS on UNIX [367](#)
- inter-process communication [217](#)
 - database storage [223](#)
 - examples [224](#)
 - overview [218](#)
 - timeout [221](#)
- iterate loop [95](#)

J

- JAAS Authentication [271](#)

K

- key
 - incoming event [54](#)
 - used for inter-process communication [218, 221](#)

L

LibraryBuilder Overview [326](#)
 Loading a User Palette [18](#)
 loops [93](#)
 accumulate output [93](#)
 index variable [93](#)
 iterate [95](#)
 repeat on error until true [101](#)
 repeat until true [98](#)
 while true [100](#)

M

main window [8](#)
 Managing Resource Conflicts in Design-time Libraries [339](#)
 mapping
 addressing schema elements [162](#)
 Input tab icons [121](#)
 XPath operators and functions [167](#)
 mappings [120](#)
 Modifying
 a ClearCase Project [365](#)
 a PVCS Project [370](#)
 ClearCase Project [365](#)
 PVCS Project [370](#)
 MTOM attachments [247](#)
 multiple events resuming a running process
 instance [225](#)

N

Nested Transactions [202](#)
 no action groups [88](#)
 Notify activity [218](#)
 Notify Configuration shared configuration [220](#)

O

opening projects [28](#)

P

palette mode [15](#)
 palettes [11](#)
 closing [17](#)
 custom [17](#)
 preferences [15](#)
 process definitions [38](#)
 breakpoints [313](#)
 breakpoints for testing [313](#)
 conditions [75](#)
 creating transitions [74](#)
 debugging [311](#), [312](#)
 grouping activities [78](#)
 handling errors [174](#)
 index variable in loops [93](#)
 new [38](#)
 process error schemas [181](#)
 process starters [65](#)
 propagating errors [179](#)
 scalability and incoming events [226](#)
 stepping through to test [318](#)
 subprocesses [43](#)
 test window [315](#)
 testing [312](#)
 ungrouping activities [79](#)
 process engines
 in operation [37](#)
 process instances
 communicating [218](#)
 enforcing order of execution [224](#)
 multiple events resumng [225](#)
 when testing [316](#)
 process modeling [34](#)
 process starters [65](#)
 process variables
 \$error [176](#)
 processes [37](#)
 project templates [24](#)

- projects 9
 - adding adapter resources 29
 - opening 28
 - saving 26
- propagation of errors 179
- PVCS Version Manager 370

R

- Receive Notification 218
- repeat on error until true loop 101
- repeat until true loop 98
- resources 10
 - unknown 11

S

- Saving
 - a Project 27
 - a Project as a Template 27
- saving projects 26
- scalability
 - incoming events 226
- schema elements 118
 - addressing 162
- schemas
 - process error schemas 181
- search predicates in XPath expressions 164
- Security Context Propagation 267
- semaphores 218
- setting a breakpoint 314
- setting breakpoints 313
- setting which palettes display 17
- shared configuration 42
- Showing Design-time Libraries 338, 338
- specifying constants in XPath expressions 126
- Start activity 67
 - subprocesses 43
- starting
 - process instances 65
- Starting TIBCO Designer 3

- starting TIBCO Designer 3
- startup options 3
- subprocesses 43, 43
 - Call Process activity 43
 - error propagation 180
 - illustrated 44

T

- templates
 - project 24
- test mode 311
- testing process definitions 311, 312
 - breakpoint locations 313
 - colors in test mode 319
 - menus and toolbar icons 320
 - overview 312
 - process instances 316
 - stepping through activities 318
 - test window 315
- TIBCO Designer roadmap 8
- TIBCO Product Documentation and Support
 - Services xxi
- TIBCO_HOME xxv
- timeout
 - incoming events 54
 - inter-process communication 221
- Transactions 185
 - Configuring Transaction Managers 193
 - JDBC 188
 - JMS Local 199
 - JTA User Transaction 189
 - Precreate XA Transactions 198
 - Recovery 196
 - XA 191
- transitions 40, 72
 - conditions 75
 - creating 74

U

- ungrouping [79](#)
- unknown resource [11](#)
- Updating a Design-time Library [336](#)
- Using
 - a Filter [109](#)
 - a Resource in the Design-time Library [337](#)
- Using Web Services Security Policies [264](#)

W

- Wait activity [218](#)
- waiting for incoming events [54](#)
- while true loop [100](#)
- Working With
 - CVS [367](#)
- WSIL files and UDDI registries [274](#)

X

- XPath [161](#)
 - basics [162](#)
 - conditions [75](#)
 - editor [165](#)
 - evaluation context [163](#)
 - example [167](#)
 - operators and functions [167](#)
 - search predicates [164](#)
 - specifying constants [126](#)
- XSLT statements [120](#), [120](#)