



TIBCO ActiveMatrix® BPM SOA Composite Development

*Software Release 4.3
April 2019*

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 OF 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 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, the TIBCO logo, Two-Second Advantage, TIB, Information Bus, ActiveMatrix, Business Studio, Enterprise Message Service, Hawk, and Rendezvous are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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. Please see the readme.txt 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.

This and other products of TIBCO Software Inc. may be covered by registered patents. Please refer to TIBCO's Virtual Patent Marking document (<https://www.tibco.com/patents>) for details.

Copyright © 2010-2019. TIBCO Software Inc. All Rights Reserved.

Contents

Figures	11
TIBCO Documentation and Support Services	12
TIBCO Business Studio	14
TIBCO Business Studio Configuration	14
Target Platforms	14
Displaying the Target Platform	14
Creating a Target Platform File	14
Setting the Target Platform	15
Platform Installation	16
Setting the Platform Installation	16
Platform Installation Reference	16
Starting TIBCO Business Studio	17
Accessing TIBCO Business Studio Help	17
SOA Projects	18
Special Folders	18
Enabling and Disabling Special Folders	18
Versions	19
WSDL Files	20
Renaming a Port Type	20
Validation	20
Validating WSDL and Schema Files Referenced by a Composite	21
Disabling Validation	21
Resolving Validation Rule Violations	22
Creating an Empty SOA Project	23
Creating a Basic SOA Project	23
Creating a SOA Project from WSDL Files	23
Creating an SOA Project from an Implementation	25
Finding out the Version of TIBCO Business Studio Using which a Project was Created	25
Enabling "TIBCO SOA Platform Extension"	28
Legacy Projects	28
Migrating Legacy Projects	28
Migration Reference	29
Composites	33
Creating a Composite	34
Setting Composite Diagram Preferences	35
Adding an Element to a Composite	35

Popup Toolbars	35
Palette	36
Application Modularization	37
Run Configuration Reference	39
Running and Debugging a Composite	41
Creating a Run Configuration	42
Running a Composite	42
Run Configuration Reference	43
Creating a Debug Configuration	45
Debugging a Composite	45
Setting Breakpoints	46
Removing Breakpoints	47
Configuring Breakpoints	47
Controlling Composite Execution	47
Debug Configuration Reference	48
Terminating a Configuration	50
Composite Reference	51
Components	55
Creating a Component	55
Changing the Implementation Type of a Component	56
Configuring a Components External Custom Feature	56
Component Reference	57
Services and References	61
Interfaces	61
Adding a Service or Reference to a Component	61
Adding a Service or Reference to a Composite	62
Promoting a Component Service or Reference	62
Static and Dynamic Wiring	62
Creating a Static Wire Between a Service and a Reference	63
Configuring a Reference for Dynamic Wiring	64
Service Reference	64
Reference Reference	66
Bindings	68
Viewing Service and Reference Bindings	69
Adding a Binding to a Service or Reference	70
SOAP Bindings	70
SOAP Binding Reference	71
Fault Messages	77
WS-Addressing	81

Endpoint References	82
Message Addressing Properties	83
Validation and Fault Handling	84
Configuring the Action Property	85
Enabling WS-Addressing	86
Generating a Concrete WSDL File	86
WSDL Generation Reference	87
Generating a SOAP Request	87
WS-Reliable Messaging	89
Enabling WS-Reliable Messaging	91
REST Bindings	91
JMS Bindings	91
JMS Binding Reference	93
Context Parameters	106
Creating Context Parameters	108
Mapping Context Parameters to Header Fields	109
Context Parameter Reference	109
Enabling Transactions in SOAP and JMS Bindings	109
Policy Management	112
Policies	112
Governed Objects	112
Intents	112
Configuring Intents	113
Intents Reference	114
At Least Once Intents	115
At Most Once Intents	115
Authorization Intents	116
Client Authentication Intents	116
Consumer Confidentiality Intents	117
Consumer Integrity Intents	117
Credential Mapping Intents	118
Managed Transaction Intents	118
Prepare Before Undeploy Intents	119
Provider Confidentiality Intents	119
Provider Integrity Intents	119
Secure Virtualization Intents	120
Start Services First Intents	120
Transacted OneWay Intents	121
Policy Sets	121

Displaying the Policy Sets Picker	122
Configuring Policy Sets	123
Editing Embedded Policy Sets	123
Creating an External Policy Set	124
Configuring External Policy Sets with XML	124
Embedded Policy Sets Reference	125
JMS At Most Once	126
JMS At Least Once	126
JMS Transacted OneWay	128
Managed Transaction	130
Message Transmission Optimization Mechanism (MTOM)	130
Prepare Before Undeploy	131
Secure Virtualization	131
Start Service First	131
Threading	131
Virtualization At Least Once	132
Virtualization Transacted OneWay	133
Virtualize	134
Never Virtualize	135
WS-Addressing for References	136
WS-Addressing for Services	136
WS-Reliable Messaging	136
Policy Templates Reference	137
Policy Template to Intents Reference	137
Authorization By Role Policies	138
Basic Authentication Policies	139
Basic Credential Mapping Policies	139
Basic Or Username Token Authentication Policies	140
SAML Authentication For SSO Policies	141
SAML Credential Mapping For SSO Policies	141
Username Token Authentication Policies	141
WS-Security Consumer Policies	142
WS-Security Provider Policies	143
Transactions	145
Managed Global Transactions	145
Transacted OneWay Transactions	146
Non-Managed Transactions	148
Resource Templates	149
Resource Instances	149

Creating a Resource Template	149
Opening a Resource Template	150
Finding References to a Resource Template	150
Renaming a Resource Template	150
Resource Templates Reference	151
Resource Templates With Scope	152
Changing the Scope of a Resource Template	154
CLI	155
Modifying Resource Templates to Install Multiple BPM Environments	156
Application Deployment Patterns	157
Hibernate	157
HTTP Client	162
JDBC	166
Configuring Third-Party JDBC Drivers	173
JMS Resource Templates	173
Configuring Third-Party JMS Drivers	174
JMS Connection Factory	174
JMS Destination	177
JNDI Connection Configuration	178
JMS Destination Configuration	181
LDAP Connection	181
Security Resource Templates	184
Identity Provider	185
Kerberos Identity Provider	186
Keystore Provider	186
Keystores	188
Creating a Keystore With a Username and Password	189
SSL Client Provider	190
SMTP	193
Thread Pool	194
Teneo	196
Properties	201
Creating a Property	202
Promoting a Component Property	203
Setting the Value of a Binding Property	203
Setting the Value of a Composite Property	203
Setting the Value of a Component Property	205
Setting the Value of a Resource Template Property	206
Creating an Obfuscated Password	208

Properties Reference	208
Substitution Variables	209
Creating a Substitution Variable	209
Creating a Substitution Variable File	210
Finding References to a Substitution Variable	210
Renaming a Substitution Variable	211
Custom Features	212
Creating a Custom Feature	212
Finding References to a Custom Feature	212
Renaming a Custom Feature	213
Configuring Plug-ins	213
Configuring Feature Dependencies	214
Configuring Excluded Custom Feature Dependencies	215
Custom Feature Reference	215
Shared Libraries	218
Bundles and Plug-in Projects	218
Creating a Library Plug-in Project	220
Distributed Application Archives	222
Packaging a Composite in a DAA	222
Packaging a Custom Feature in a DAA	223
Regenerating a Distributed Application Archive Containing a Composite	223
Generating and Verifying Derived DAAs	224
Distributed Application Archive Specifications	224
Qualifier Replacement	224
Distributions	226
Creating a Distribution	226
Editing a Distribution	227
Creating a Distribution Template	228
Applications	229
Creating, Deploying, and Starting an Application	230
Stopping, Undeploying, and Deleting an Application	231
Debugging Deployed Applications	231
Checking Remote Debugger Status	232
Creating a Debug Configuration	232
Launching a Debug Configuration	233
Attaching a Debugger to a Running Application	233
TIBCO ActiveMatrix Administrator	235
Deployment Servers	235
Creating a Deployment Server	235

Connecting to a Deployment Server	235
Editing Deployment Server Properties	236
Refreshing a Deployment Server	236
Disconnecting from a Deployment Server	236
Deployment Server Reference	236
Administrator Explorer View	237
Displaying the Administrator Explorer View	239
Administrator Command-Line Interface	239
Understanding Build Files	240
Understanding AMXAdminTask	244
Actions Performed Using CLI	251
Understanding Data Files	252
Understanding Objects	254
Object Formats	254
Property File Reference	255
Generating an Administrator Command-Line Interface Script	257
Script Configuration	257
Administrator Connection Configuration	258
Application Configuration	259
Distribution	260
Property Configuration	261
Wiring Configuration	263
Running Administrator Command-Line Interface Scripts	263
TIBCO Business Studio Command-Line Interface	264
Running a TIBCO Business Studio Command-Line Interface Script	264
TIBCO Business Studio Command-Line Reference	266
TIBCO Business Studio Command-Line Task Reference	266
Logging	281
Application Logging	281
Message Flow Logging	284
Logging Payload Reference	291
IPv6 Support	292

Figures

Composite	33
Composite Editor	34
Palette	36
Modularization Options	38
Static and Dynamic Wiring	63
Bindings	69
Bindings in Business Studio	69
WS-RM Participants	89
Reliable Messaging Participants	90
Service and Reference Request and Reply Communication	92
Managed Global Transaction Flow	145
OneWay Transaction	147
Non-managed Transaction Flow	148
TIBCO Business Studio Resource Instance	149
ActiveMatrix Administrator Resource Instance	149
Properties	202
Application Life Cycle	229
Administrator Explorer View	238
HelloWorld Message Flow	285
Hello World and Date Manager Message Flow	287

TIBCO Documentation and Support Services

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the TIBCO Product Documentation website, mainly in 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>.

Product-Specific Documentation

Documentation for TIBCO ActiveMatrix® Service Grid is available on the <https://docs.tibco.com/products/tibco-activematrix-service-grid> page.

Use of the following features, installation profiles and development tools requires a TIBCO ActiveMatrix Service Grid license:



- TIBCO ActiveMatrix Policy Director Governance, TIBCO ActiveMatrix SPM Dashboard, and TIBCO ActiveMatrix SPM Runtime Server profiles; and
- TIBCO ActiveMatrix Service Grid development tools for Java, Webapp and Spring components.

Customers with only a TIBCO ActiveMatrix Service Bus license are not licensed to use these features, tools or profiles.

The following documents form the documentation set:

- *TIBCO ActiveMatrix Service Grid Concepts*: Read this manual before reading any other manual in the documentation set. This manual describes terminology and concepts of the platform. The other manuals in the documentation set assume you are familiar with the information in this manual.
- *TIBCO ActiveMatrix Service Grid Development Tutorials*: Read this manual for a step-by-step introduction to the process of creating, packaging, and running composites in TIBCO Business Studio.
- *TIBCO ActiveMatrix Service Grid Composite Development*: Read this manual to learn how to develop and package composites.
- *TIBCO ActiveMatrix Service Grid Java Component Development*: Read this manual to learn how to configure and implement Java components.
- *TIBCO ActiveMatrix Service Grid Mediation Component Development*: Read this manual to learn how to configure and implement Mediation components.
- *TIBCO ActiveMatrix Service Grid Mediation API Reference*: Read this manual to learn how to develop custom Mediation tasks.
- *TIBCO ActiveMatrix Service Grid Spring Component Development*: Read this manual to learn how to configure and implement Spring components.
- *TIBCO ActiveMatrix Service Grid WebApp Component Development*: Read this manual to learn how to configure and implement Web Application components.
- *TIBCO ActiveMatrix Service Grid REST Binding Development*: Read this manual to learn how to configure and implement REST components.
- *TIBCO ActiveMatrix Service Grid Administration Tutorials*: Read this manual for a step-by-step introduction to the process of creating and starting the runtime version of the product, starting TIBCO ActiveMatrix servers, and deploying applications to the runtime.
- *TIBCO ActiveMatrix Service Grid Administration*: Read this manual to learn how to manage the runtime and deploy and manage applications.

- *TIBCO ActiveMatrix Service Grid Hawk ActiveMatrix Plug-in*: Read this manual to learn about the Hawk plug-in and its optional configurations.
- *TIBCO ActiveMatrix Service Grid Policy Director Governance Custom Actions*: Read this manual to learn how you can configure and enforce policies for ActiveMatrix and external services hosted in third party containers, using TIBCO ActiveMatrix Policy Director Governance.
- *TIBCO ActiveMatrix Service Grid Service Performance Manager API Reference*: Read this manual to learn how to use the SPM APIs.
- *TIBCO ActiveMatrix Service Grid Error Codes*: Read this manual to know more about the error messages and how you could use them to troubleshoot a problem.
- *TIBCO ActiveMatrix Service Grid Installation and Configuration*: Read this manual to learn how to install and configure the software.
- *TIBCO ActiveMatrix Service Grid Security Guidelines*: Read this manual to learn more about security guidelines and recommendations for TIBCO ActiveMatrix Service Grid.
- *TIBCO ActiveMatrix Service Grid Release Notes*: Read this manual for a list of new and changed features, steps for migrating from a previous release, and lists of known issues and closed issues for the release.

How to Contact TIBCO Support

You can contact TIBCO Support in the following ways:

- For an overview of TIBCO Support, visit <http://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](https://community.tibco.com). For a free registration, go to <https://community.tibco.com>.

TIBCO Business Studio

TIBCO Business Studio(TM) provides a common modeling, implementation and deployment environment for different types of applications. TIBCO Business Studio provides an Eclipse-based design environment which:

1. Business analysts can use to capture, design and model all aspects of a business process, including the organization and data models that underpin it.
2. Solution designers can implement the process as an executable application, then deploy the application to the TIBCO ActiveMatrix runtime for execution.

Applications developed using TIBCO ActiveMatrix design environment conform to Service Component Architecture (SCA). SCA is a model for developing applications based on the service-oriented architecture (SOA).

TIBCO Business Studio Configuration

TIBCO Business Studio supports configuration in the **Window > Preferences** property sheets. The subsections most relevant to TIBCO Business Studio are **General > Web Browser** , **Diagram, Plug-in Development > Target Platform** , and **TIBCO SOA Platform**.

Target Platforms

A *target platform* is the set of platform plug-ins against which you develop and test user-defined plug-ins. In TIBCO ActiveMatrix, user plug-ins contain custom mediation task implementations and shared libraries. The target platform encompasses the folders containing the plug-ins, the list of plug-ins, the target environment, and launching arguments. There is one target platform per workspace and it applies to all projects in the workspace. By default, the target platform is set to *TIBCO_HOME/components/shared/1.0.0*.

A *target platform file* is an XML file that encapsulates a target platform (location, content, environment, and so on). It defines a set of the bundles contained in the location identified by the TIBCO Home field against which the bundles in the workspace are compiled and run.

See also **Help > Help Contents > Supplemental Eclipse Help > Plug-in Development Environment Guide > Reference > Preferences > Target Platform Preferences** .

Displaying the Target Platform


Procedure

1. Select **Window > Preferences**.
2. Click **Plug-in Development > Target Platform** node.
The target platform editor displays. In the Target definitions list, the active target platform is bolded and identified by a checked checkbox and the string (Active). If you select a target platform, the folder containing the target platform plug-ins displays in the Locations field.

Creating a Target Platform File

Procedure

1. **File > New > TIBCO SOA Resources** .
The TIBCO SOA Resource Wizard displays.
2. Click **Target Platform** and click **Next**.
The New Target Platform File dialog displays.


3. Type a folder name in the Enter or select the parent folder field or select a folder in the folder list.
4. In the File name field, accept the default name or type a new one and click **Next**.
5. In the Target Name field, accept the default name or type a new one.
6. In the TIBCO Home field, accept the default or click  to select another TIBCO Home location.
7. In the Product Name list, check the checkboxes next to the product plug-ins to include in the target platform. The default selection is TIBCO ActiveMatrix Platform.
8. Click **Finish**.

Setting the Target Platform

You can set the target platform using a predefined target platform source or a target platform file.

Procedure

- Choose a target platform source and follow the appropriate procedure.

Target Platform Source	Procedure
Predefined	<ol style="list-style-type: none"> 1. Select Window > Preferences. 2. Click the Plug-in Development > Target Platform node. 3. If the target platform you want is not listed in the Target definitions list, click Add...: <ol style="list-style-type: none"> a. Click the Template radio button, click , and select one of the following: <ul style="list-style-type: none"> • TIBCO ActiveMatrix SOA Studio - Target platform for plug-ins that will be executed in TIBCO Business Studio. • TIBCO ActiveMatrix Runtime - Target platform for plug-ins that will be executed in the TIBCO ActiveMatrix runtime environment. b. Click Next. c. Click Finish. 4. Check the checkbox next to the desired predefined target definition. 5. Click Apply. 6. Click Reload. 7. Click OK.
Target Platform File	<ul style="list-style-type: none"> • Project Explorer <ol style="list-style-type: none"> 1. In the Project Explorer, double-click a target platform file. 2. In the top-right of the target platform file editor, click Set as Target Platform. • Environment variable - Define the environment variable <code>com.tibco.target.platform.location</code> and set it to the absolute path to the target platform file. • TIBCO Business Studio configuration

Target Platform Source	Procedure
	<ol style="list-style-type: none"> 1. In <code>TIBCO_HOME/studio/version/eclipse/TIBCOBusinessStudio.ini</code>, specify <code>-Dcom.tibco.target.platform.location=absolute path to the .target file</code>. 2. Restart TIBCO Business Studio. <ul style="list-style-type: none"> • Command line <ol style="list-style-type: none"> 1. Specify <code>-Dcom.tibco.target.platform.location=absolute path to the .target file</code> when starting TIBCO Business Studio from the command line.

Platform Installation

Multiple versions of a TIBCO ActiveMatrix product can be installed into the same `TIBCO_HOME` folder. The platform installation allows you to set the location of `TIBCO_HOME` folder and choose the version of the TIBCO Host that will be used when you run a composite in a local debug node and the version of Administrator that will be used when you generate an Administrator CLI script.

Setting the Platform Installation

Procedure

1. Select **Windows > Preferences** .
The Preferences dialog displays.
2. Click **TIBCO SOA Platform > Platform Installation** .
The Platform Installation screen displays.
3. In the TIBCO Home Location field, click the **Browse** button.
The Browse for Folder dialog displays.
4. Click a `TIBCO_HOME` folder and click **OK**.
The TIBCO Host Version and Administrator Version fields are updated with the versions available in the selected `TIBCO_HOME`.

Platform Installation Reference

Field	Description
TIBCO Home Location	<p>The location of the ActiveMatrix product installation.</p> <p>Default: The value of the system properties <code>com.tibco.target.platform.location</code> or <code>TIBCO_HOME_LOCATION</code>. If neither of the properties are defined, the location of the TIBCO Business Studio or the target platform setting.</p>
TIBCO Host Version	TIBCO Host version that will be used when you run a composite in a local debug node.

Field	Description
Platform Version	<p>The hotfix or GA version of the ActiveMatrix platform. Hotfixes installed in the selected TIBCO_HOME appear as options in the drop-down.</p> <p>If the Latest option is selected, the latest installed GA version or hotfix is used. You need not change the version for every installation.</p> <p>Default: Latest</p>
Administrator Version	Version of Administrator that will be used when you generate an Administrator CLI script.

Starting TIBCO Business Studio

You can start TIBCO Business Studio on a Windows command prompt from the *TIBCO_HOME* directory.

- **Windows** - Run *TIBCO_HOME\studio\version\eclipse\TIBCOBusinessStudio.exe*
- **UNIX** - Run *TIBCO_HOME/studio/version/eclipse/TIBCOBusinessStudio*

Accessing TIBCO Business Studio Help

By default, documentation access from TIBCO Business Studio™ is online, through the TIBCO Product Documentation site (Doc site) at <https://docs.tibco.com/> which contains the latest version of the documentation. You can access online help using **Help > TIBCO Business Studio SOA Edition Help Content** option. To access the product documentation offline, download the documentation to a local directory and then change the help preferences in TIBCO Business Studio as described in the following steps:

Procedure

1. Go to the **Help** menu.
2. Select **Download Offline TIBCO Help...**
3. In Content access strategy section, by default **Prefer offline help (if available)** option is selected but online help will be used until help is downloaded. The alternative selection is **Always use online help**.
4. To define where you want the offline help to be located, browse to the location in which you want to save the Base offline documentation folder and click **Apply**. You can click **Location** to go to the location that is defined.
5. If you have not already downloaded the documentation, click **Download** button in right side to download documentation from docsite to the location specified in the **Base offline documentation** field. If the **Prefer offline Help (if available)** content access strategy is selected then the downloaded help content will be used.

SOA Projects

In TIBCO Business Studio you develop SOA assets in SOA projects using SOA project wizards.

To support rapid development of commonly used scenarios, TIBCO Business Studio provides four SOA project wizards:

- **Creating an Empty SOA Project** - Creates an SOA project with an empty composite.
- **Creating a Basic SOA Project** - Creates an SOA project, composite, abstract component with a service and reference, and promoted composite service and reference. The services and references are configured with a sample WSDL file.
- **Creating a SOA Project from WSDL Files** - Creates an SOA project, composite, component and promoted services, optional component and promoted references, and component implementation with services and optional references specified by port types selected from one or more WSDL files.
- **Creating an SOA Project from an Implementation** - Creates an SOA project with a composite, a component, and one or more promoted services and references from an existing implementation.

Special Folders

During SOA project creation you have the option to create special folders to contain SOA asset types.

The following SOA asset types are supported:

- **Service Descriptors** - Service descriptors such as WSDL and schema files
- **Composites** - Composite files
- **Mediation Flows** - Mediation flows
- **Resource Templates** - Resource template, substitution variable, and policy set files
- **Deployment Artifacts** - Distributed application specification and archive, distribution, custom feature, and command-line interface build and property files

Special folders created in an SOA project have the following features:

- Asset-specific behavior in the Project Explorer. For example, in the **Service Descriptors** folder you can expand a WSDL file to view its constituent operations and messages.
- When you right-click a special folder and select **New**, the appropriate asset type will be available in the context menu.

Enabling and Disabling Special Folders

You can enable a normal folder as a special folder, and you can disable special folder capabilities.

Procedure

- Choose an action and follow the appropriate procedure.

Action	Procedure
Enabling	1. Right-click a normal folder and select Special Folders > Use as X , where X is Service Descriptors, Composites, Mediation Flows, Resource Templates, or Deployment Artifacts.

Action	Procedure
Disabling	1. Right-click a special folder and select Special Folders > Do not use as X Folder , where X is Service Descriptors, Composites, Mediation Flows, Resource Templates, or Deployment Artifacts.

Versions

A *version* is a property that controls how an object is treated at installation or deployment. Versions are specified in TIBCO Business Studio and cannot be modified in Administrator.

The following objects have versions:

- Composites and application templates.
- Components - During application upgrade, Administrator compares component versions to determine whether the component needs to be upgraded.
- Features
- Plug-ins
- Packages

Version Numbers

A version number is a multicomponent number of the form *major.minor.service.qualifier*. Changes in the value of each component reflect different types of changes in the versioned object:

- *major* - Reflects breaking changes to the interface.
- *minor* - Reflects non-breaking changes in an externally visible way. Examples of externally visible changes include binary compatible changes, significant performance changes, major code rework, and so on.
- *service* - Reflects changes that are not visible in the interface. For example, a bug has been fixed in the code, documentation has changed, compiler settings have changed, and so on.
- *qualifier* - Identifies when and where the object was built or packaged.

When you create an object in TIBCO Business Studio, the version is set to "1.0.0.qualifier". If the *qualifier* component of a version is set to "qualifier" when you create a DAA, TIBCO Business Studio replaces "qualifier" with a generated qualifier that defaults to a timestamp. You can customize the format of the generated qualifier by specifying a qualifier replacement.

Version Ranges

Some fields require you to specify a version range. For example, a feature may have a dependency on a range of versions of another feature. A *version range* is an interval specified as: *bracket lower limit, upper limit bracket*, where *bracket* can be "[" or "]", which denotes an inclusive end of the range or "(" or ")", which denotes an exclusive end of the range. If one end of the range is to be included and the other excluded, the range can contain a square bracket with a round bracket.

There are three common use cases:

- A strict version range, such as [1.0.0,1.0.0], denotes version 1.0.0 and only that version.
- A half-open range, such as [1.0.0,2.0.0), which has an inclusive lower limit and an exclusive upper limit, denotes version 1.0.0 and any later version, up to, but not including, version 2.0.0.
- An unbounded open range expressed as a single number such as 2.0.0, which is equivalent to the range [2.0.0, infinity), denotes version 2.0.0 and any later version.

WSDL Files

TIBCO ActiveMatrix services are described using WSDL port types.

TIBCO ActiveMatrix supports services that comply with the [WSDL 1.1 specification](#).



Importing schema types using `xsd:import` is not supported. Instead, embed the types or import using `wsdl:import` inside the `wsdl:definitions` element.




If you change the order of operations in the WSDL interface of a service or reference you must recreate all SOAP bindings associated with the service or reference.

Renaming a Port Type

You can change the port type for a WSDL file in the WSDL editor.

Procedure

1. Open a WSDL file in the WSDL Editor.
2. Click the port type.
3. Click the Rename icon () to the right of the Name field.
4. In the New name field, type a new name.
5. Click **Preview**.
6. Uncheck the checkboxes next to the changes you don't want performed.
7. Click **OK**.


Validation

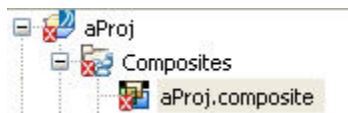
TIBCO Business Studio validates the information that you provide when you create and configure composite elements. Validation occurs while you create objects and enter values in wizards and property sheets.

When a validation rule is violated TIBCO Business Studio can issue a warning or an error. When possible, TIBCO Business Studio provides quick fixes that suggest one or more ways to resolve the error.

Warnings identify invalid information but usually do not imply that future actions on the component are restricted. In contrast, errors may restrict future actions on the component. For example, a service assembly flagged with an error may not be deployable.

When TIBCO Business Studio issues a warning or error, one of the following may occur:

- The composite element that violates a validation rule and every container (composite, SOA project) containing that element and is marked with an error badge 



- Properties view - The property containing the error is marked with an error badge:

Application Library

Library Name:


Library Path:

- Problems view - A warning or error is added to the Problems tab:

Properties Problems Registries			
1 error, 1 warning, 0 infos			
Description	Resource	Path	Location
Errors (1 item)			
Implementation Script for 'Composite ref/Component1/Script Implementation Ruby ref.composite	ref.composite	ref/Composites	//@servi...
Warnings (1 item)			
WS-I: (BP2120) A binding has operations that are not unique.	LoanInterfa...	ref/Composites	line 58

- Wizard - Either the Next button continues to be disabled or the wizard displays a popup describing the error. For example:

Project

 . is an invalid name on this platform.

Validation Rule Examples

- Component references must be wired to component services or promoted references.
- Component services must be wired to component references or promoted services.
- The WSDL interface of component references and services must be configured.
- Component configurations must match their implementations.
- All components except for abstract components must be configured with an implementation.

Validating WSDL and Schema Files Referenced by a Composite

WSDL and schema files referenced by composite files are validated by default. If you disable validation, you can manually validate WSDL and schema files.

- In the Project Explorer view, right-click a composite and select **Validate Related WSDL/XSD**. The WSDL and schema files referenced by the composite are validated.

Disabling Validation

You can disable validation to speed up operation execution time using the command-line interface.

The automatic validation the WSDL and schema files referenced by a composite performed by TIBCO Business Studio can consume a significant portion of the time to perform operations involving a composite. You can disable validation to speed up importing a large set of projects using the TIBCO Business Studio command-line interface.

Procedure

- Choose the operations for which validation should be disabled and follow the appropriate procedure.

Validation Disabled For	Procedure
All operations	<ul style="list-style-type: none"> Specify – <code>Dsoa.disableWSDLValidation=true</code> when starting TIBCO Business Studio. 1. Select Windows > Preferences. 2. Select TIBCO SOA Platform > WSDL Validation. 3. Check the Disable Live WSDL Validation checkbox. Click OK.
All operations except DAA creation	<ol style="list-style-type: none"> 1. Select WindowsPreferences. 2. Select TIBCO SOA Platform > WSDL Validation. 3. Check the Disable WSDL Validation checkbox. 4. Click OK.

Resolving Validation Rule Violations

The UI supports different validation procedures for rule validation for different error messages.

Procedure

- Follow the appropriate procedure to resolve the error message.

Error Message	Procedure
Component and Implementation Out of Sync	<ol style="list-style-type: none"> 1. Right-click the component, select Quick Fixes and then select Update Component from Implementation or Update Implementation. 2. Right-click an error in the Problems view, select Quick Fix and then select Update Component from Implementation or Update Implementation. Click Finish. 3. Hover over the error badge in the composite and select Update Component from Implementation or Update Implementation.
WSDL Interface Not Configured	<ol style="list-style-type: none"> 1. Right-click the reference and select Select Port Type. 2. Click the port type. 3. Click OK.
Component Reference Not Wired	<ol style="list-style-type: none"> 1. Promote the reference or wire to another component service.

What to do next

Sometimes the validator will not completely refresh the workspace even after you have resolved the errors. To manually refresh the workspace, select **Project > Clean** and click **OK**.

Creating an Empty SOA Project

An empty SOA project includes only an empty composite and no other elements.

Procedure

1. Select **File > New > TIBCO SOA Resources** .
2. Click **TIBCO SOA Project** and click **Next**.
3. In the Project name field, accept the default name or type a new name and click **Next**.
4. Uncheck the checkboxes next to any asset type you don't want created and click **Next**.
5. In the Project Types list, choose **Empty SOA Project** and click **Next**.
6. In the Composite File field, accept the default name or type a new name.
7. Optionally rename the special folders and click **Finish**.

Result

The system creates an SOA project with an empty composite.

Creating a Basic SOA Project

A basic SOA project includes a composite, abstract component with a service and reference, and promoted composite service and reference.

Procedure

1. Select **File > New > TIBCO SOA Resources** .
2. In the Project Types list, click **Basic SOA Project** and click **Next**.
The Composite Details dialog displays.
3. Accept the default names or type new names for the composite file, component, service, and reference and click **Next**.
4. Optionally, rename the special folders and click **Finish**.

Result

The system creates an SOA project, composite, abstract component with a service and reference, and promoted composite service and reference. The services and references are configured with a sample WSDL file.

Creating a SOA Project from WSDL Files

You can create a SOA project from WSDL files

Prerequisites

Obtain one or more WSDL files.

Procedure

1. Select **File > New > TIBCO SOA Resources** .
2. In the Project Types list, click **SOA Project From WSDL** and click **Next**.
The Service Port Types dialog displays.

3. In the Composite File field, accept the default name or type a new name.
4. In the Selected Port Types field, select a port type and click **Next**, or click **Finish** if you do not want to continue.

Option	Procedure
Import new WSDL file	<ol style="list-style-type: none"> 1. Click Import. The Select External WSDL dialog displays. 2. Browse to a folder containing a WSDL file, click the file, and click Open. 3. The port type contained in the file is added to the Selected Port Types list.
Add port type from workspace	<ol style="list-style-type: none"> 1. Click Add. The Select Port Types dialog displays. 2. Click a port type and click Add. 3. Click OK. 4. The port type is added to the Selected Port Types list.

A composite is created with an abstract component and a promoted service.

5. In the Service Binding Details screen, check the checkbox next to one or more bindings and optionally click in the Binding Name column, rename the bindings, and click **Next**, or click **Finish** if you do not want to continue.



In SOAP/HTTP Binding, you can use substitution variables for the fields **Connector Name** and **HTTP Client** corresponding to a SOAP service binding or a SOAP reference binding.

A composite is created with an abstract component and a promoted service with the specified bindings and the Reference Port Types dialog displays.

6. Optionally select a port type and click **Next**.

Option	Procedure
Import new WSDL file	<ol style="list-style-type: none"> 1. Click Import. The Select External WSDL dialog displays. 2. Browse to a folder containing a WSDL file, click the file, and click Open. 3. The port type contained in the file is added to the Selected Port Types list.
Add port type from workspace	<ol style="list-style-type: none"> 1. Click Add. The Select Port Types dialog displays. 2. Click a port type and click Add. 3. Click OK. 4. The port type is added to the Selected Port Types list.

7. Check the checkbox next to one or more bindings and optionally click the Binding Name column and rename the bindings and click **Next**, or click **Finish** if you do not want to configure component details.

A composite is created with an abstract component and promoted service and reference with the specified bindings.

8. Click **Next**.
The Component Details screen displays.
9. Configure the component.
 - a) In the Component Name field, accept the default name or type a new one.

- b) Click a component implementation type and click **Next** for all component implementation types except Abstract.
See the implementation type documentation for implementation-specific steps.
- c) Click **Finish**.

Result

The system creates a SOA project, composite, component and promoted services, optional component and promoted references, and component implementation with services and optional references specified by port types selected from one or more WSDL files.

Creating an SOA Project from an Implementation

You can create a SOA project from an already existing implementation. The resulting project includes a composite, abstract component with a service and reference, and promoted composite service and reference.

Prerequisites

Import the package containing the component implementation into the workspace.

Procedure

1. Select **File > New > TIBCO SOA Resources**.
2. In the Project Types list, click **SOA Project from Implementation** and click **Next**.
The Component Details dialog displays.
3. In the Composite File field, accept the default composite file name or type a new one.
4. In the Component Name field, accept the default component name or type a new one.
5. Click a component implementation type and click **Next**.
6. Refer to the implementation type documentation for implementation-specific steps and click **Next**.
7. Select a port type.
8. Check the checkbox next to a binding and optionally rename the binding and click **Finish**.

Result

The system creates a SOA project, composite, abstract component with a service and reference, and promoted composite service and reference. The services and references are configured with a sample WSDL file.

Finding out the Version of TIBCO Business Studio Using which a Project was Created

You can find out the version of TIBCO Business Studio using which a project was created. This information is stored in a `.version` file.

A `.version` file is created when:

- A new project is created. The `.version` file can be seen under **Service Descriptors** in the Project Explorer.
- A Java Implementation is generated. The `.version` file can be seen under **META-INF** in the Project Explorer.
- Projects are imported. The `.version` file can be seen under **META-INF** in the Project Explorer.

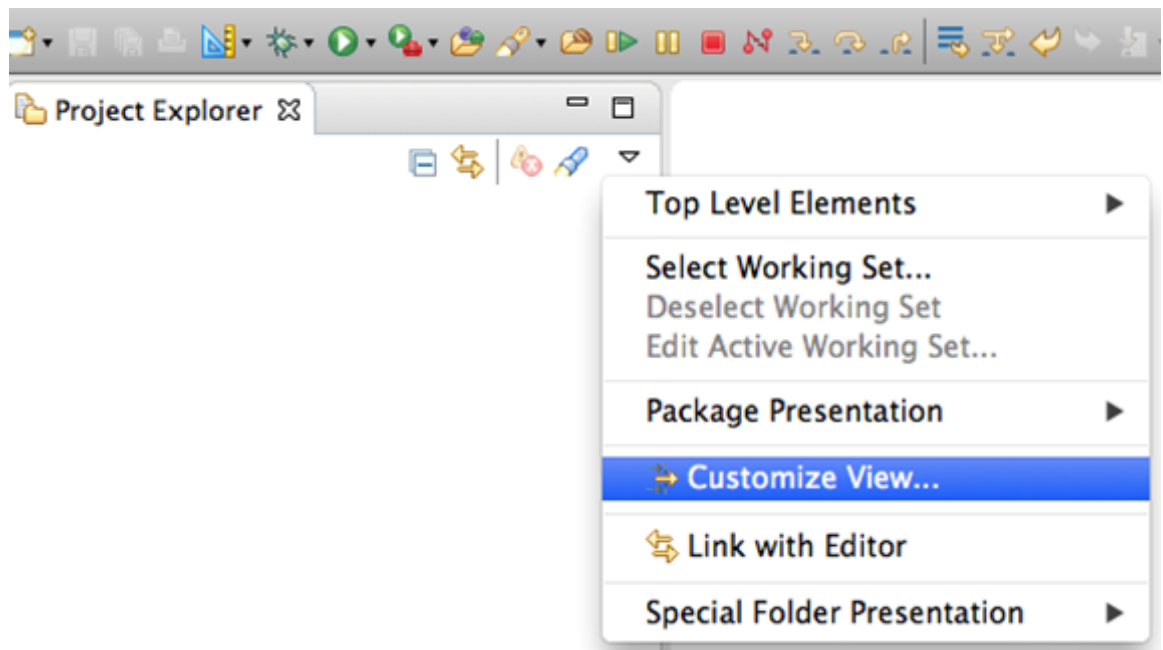
- If **Project > Build Automatically** is enabled, the `.version` file is generated.
- If **Project > Build Automatically** is disabled, the `.version` file is not generated.
- When you clean a project using the **Projects > Clean**.
 - **Clean projects selected below** option

If the selected SOA project has a reference to a Java project, the `.version` file is created on the referenced Java project even if the Java project is not selected on the Clean projects view.
 - **Clean all projects** option

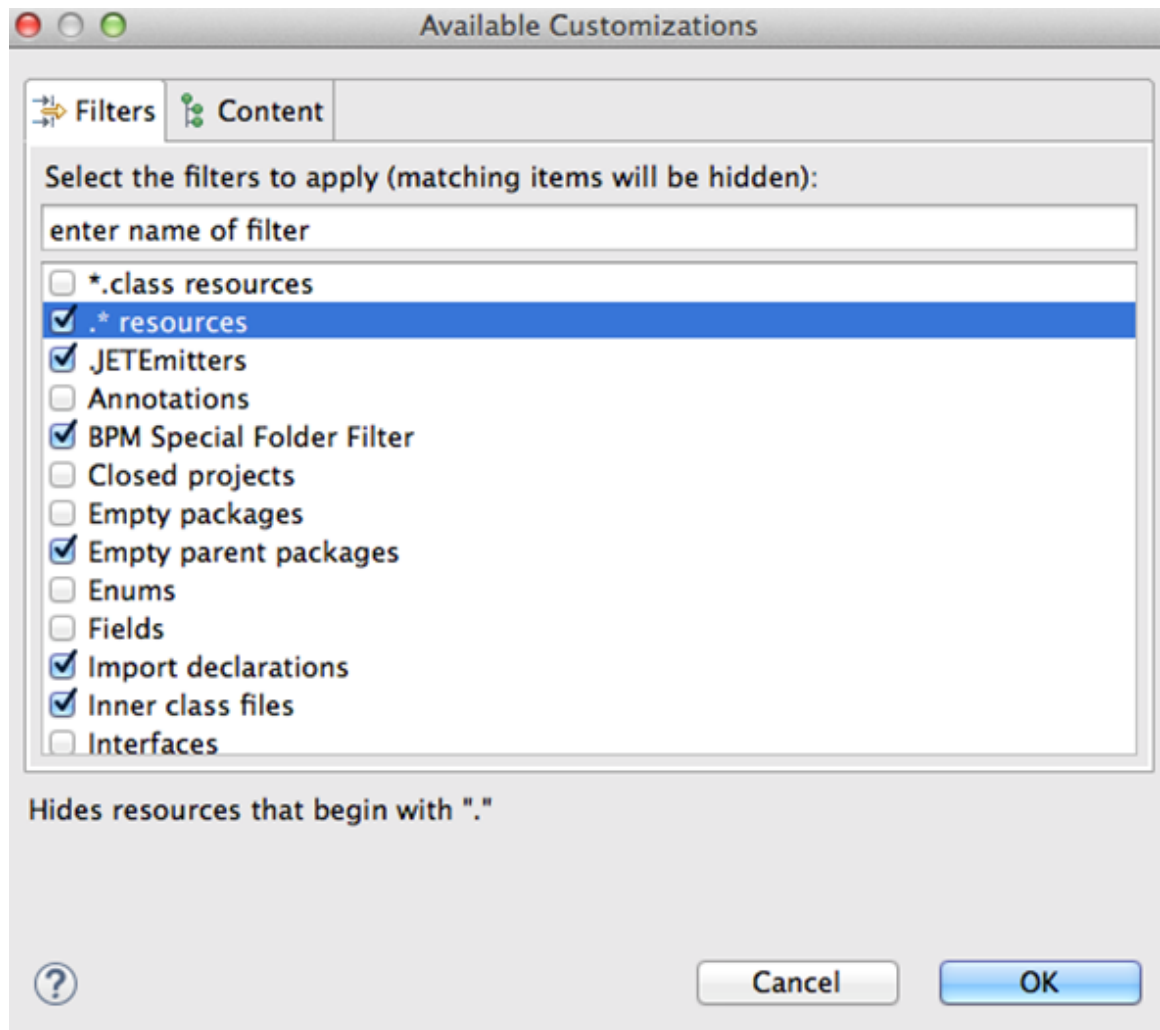
When this option is selected, it generates the `.version` file for all SOA projects (and their reference projects) in the workspace that do not have a version file.

Viewing the `.version` File

1. In the Project Explorer View, select **Customize View**. The Available Customizations dialog box appears.

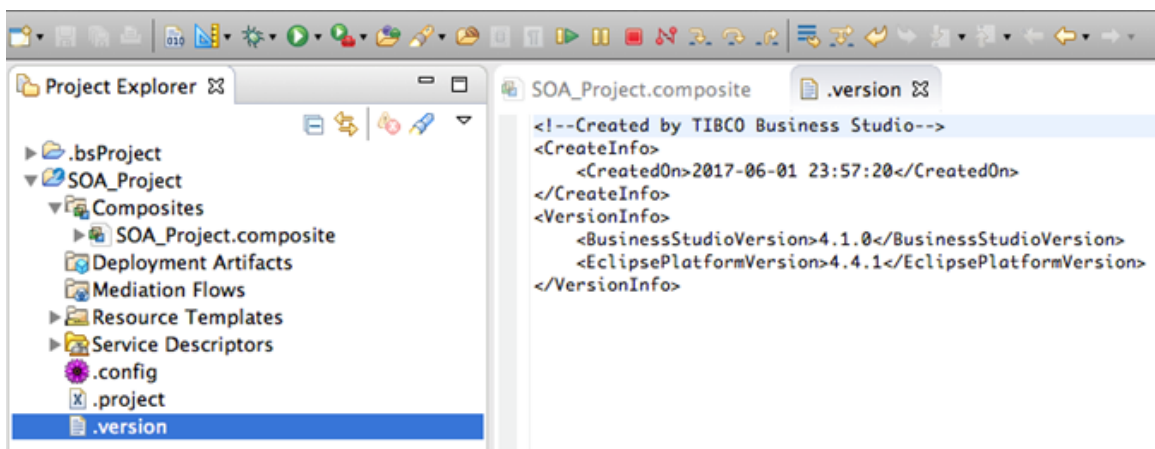


2. Uncheck `.*resources` from the list of filters displayed.



3. Click **OK**.
4. Refresh the project. You must refresh each project in Studio to see the `.version` file.

A `.version` file is generated in all the cases mentioned above. For example, when a new SOA project is created, the `.version` file is created under **Service Descriptors** as follows:



Enabling "TIBCO SOA Platform Extension"

Some Implementation Types and Binding Types require "TIBCO SOA Platform Extension" to enable their configuration. For example, the Web Application Implementation Type requires "TIBCO SOA Platform Extension" to be enabled for locating and overriding the `.requirements` file.

To enable "TIBCO SOA Platform Extension":

1. Enable **TIBCO SOA Platform** preferences.
2. On the toolbar, click the drop-down next to the **Enable/Disable Business Studio Capabilities** icon.
3. Ensure the **TIBCO SOA Platform Extension** capability is selected.

Legacy Projects

TIBCO Business Studio supports migrating TIBCO ActiveMatrix 2.x projects to TIBCO ActiveMatrix 3.x format.

The structure of TIBCO ActiveMatrix SOA projects and the resources contained within the projects changed between TIBCO ActiveMatrix 2 and TIBCO ActiveMatrix 3.



The migration process updates the type of folders but does not rename folders. For example, the folder that contains resource templates is named Shared Resources in TIBCO ActiveMatrix 2 and Resource Templates in TIBCO ActiveMatrix 3.

Migrating Legacy Projects

You can migrate a legacy project by importing it into TIBCO Business Studio.

You should import projects that are dependent on one another (for example, a TIBCO SOA project and the Java plug-in project on which it depends) at the same time.

Procedure

1. Import the legacy project into TIBCO Business Studio. For information on how to import projects, see **Help > Help Contents > Workbench User Guide > Tasks > Importing**. The Problems view displays several errors.
2. Invoke the Project Migration wizard:
 - In the Project Explorer, right-click the project to be migrated or its contained composite file and select **Migrate to ActiveMatrix 3.0 format**.
 - In the Problems view, perform the following steps:
 1. Right-click the problem and select **Quick Fix**.
 2. Click **Migrate legacy project** and click **Finish**.
3. Optionally click the add icon () to specify additional projects.
4. Optionally check the **Save a copy of these projects before migrating** checkbox and click the Browse icon () to browse to the location in which to save the projects. Saving a copy is strongly recommended, especially if not using source control.
5. Click **Next**. A tree containing the selected projects and the proposed migration actions displays.
6. Uncheck checkboxes next to the migration actions you do not want performed.

Deselect only in cases where you really do not want some resource migrated as partial migration is not supported. Resources not migrated at the same time may not be able to be correctly migrated at a later stage if they contain inbound or outbound references to other resources.

7. Click **Finish**.

The selected migration actions are performed. A summary of the migration displays in a pop-up dialog and is stored in the file *ProjName/migration.txt*, which is opened in a text editor. The summary lists the actions that were performed, the objects that were not migrated, and the manual actions that you must perform to run the migrated project.

8. Click **OK** to dismiss the dialog.

Migration Reference

Depending on your input object, you might have different migration output elements and different required actions.

Project and Component Migration

Input Object	Migration Output	Required Actions
SOA Project	Composites, Shared Resources, and Deployment Packages migrated to the correct special folder type, but folders are not renamed.	None.
Topic	Abstract component	<ul style="list-style-type: none"> Change the type of the abstract component to a valid component and implement as appropriate.

Service and Reference Migration

Input Object	Migration Output	Required Actions
SOAP/HTTP Service	<ul style="list-style-type: none"> Promoted service with SOAP binding If transport type is HTTPS, transport type is HTTP Connector Name property set to resource profile name or to httpConnector. If Enable WS-Addressing and Anonymous checked in binding, the binding is configured with policy set "WSAddressingPolicyForService_BindingName" If Enable WS-Reliable Messaging checked in binding, the binding is configured with policy set "WSRMPolicy_BindingName" 	<ul style="list-style-type: none"> Create an HTTP Connector and install instance on node prior to deployment. If transport type was HTTPS, configure HTTP Connector for SSL.

Input Object	Migration Output	Required Actions
SOAP/JMS Service	<ul style="list-style-type: none"> Promoted service with SOAP binding JMS Connection Factory, JMS Connection Factory Configuration, JMS Destination, JNDI Connection Configuration 	None.
SOAP/HTTP Reference	<ul style="list-style-type: none"> Promoted reference with SOAP binding If the binding pointed to an HTTP Client Shared Resource, an HTTP Client. 	None.
SOAP/JMS Reference	<ul style="list-style-type: none"> Promoted reference with SOAP binding JMS Connection Factory, JMS Destination (inbound), JMS Destination (outbound), JNDI Connection Configuration 	None.
SOAP Reference with Third-Party Binding	The project will migrate but will result in a composite that is not supported.	<ul style="list-style-type: none"> Convert reference to a supported binding type.
AMX Service	<ul style="list-style-type: none"> Promoted service with Virtualization binding. If a port type was set on the service then the name for the generated Virtualization binding will match the name of the port type. Otherwise, the name of the Virtualization binding will be Virtualization. 	None.
AMX Reference	<ul style="list-style-type: none"> Promoted reference with Virtualization binding. If a port type was set on the reference then the name for the generated Virtualization binding will match the name of the port type. Otherwise, the name of the Virtualization binding will be Virtualization. 	<ul style="list-style-type: none"> Wire the applications at deploy time using Administrator or create a wrapping composite with components of type composite wired together.

Input Object	Migration Output	Required Actions
JMS Service	<ul style="list-style-type: none"> • Promoted service with JMS binding • JMS Connection Factory Configuration • JMS Connection Factory • JMS Destination Configuration • JMS Destination • JNDI Connection Configuration • JMS target address is mapped to JMS Destination Configuration • JMS reply address is mapped to JMS Destination 	<ul style="list-style-type: none"> • Configure generated JMS resource templates to point to JNDI Connection Configuration.
JMS Reference	Promoted reference with JMS binding <ul style="list-style-type: none"> • Promoted service with JMS binding • JMS Connection Factory Configuration • JMS Connection Factory • JMS Destination Configuration • JMS Destination • JNDI Connection Configuration • JMS target address is mapped to JMS Destination Configuration • JMS reply address is mapped to JMS Destination 	<ul style="list-style-type: none"> • Configure generated JMS resource templates to point to JNDI Connection Configuration.

Shared Resource Migration

Input Object	Migration Output	Required Actions
JMS Shared Resource	<ul style="list-style-type: none"> • JMS Connection Factory • JMS Connection Factory Configuration • If connection type is JNDI, a JNDI Connection Configuration 	<ul style="list-style-type: none"> • If connection type is direct, create a JNDI Connection Configuration. • Configure generated JMS resource templates to point to JNDI Connection Configuration.

Input Object	Migration Output	Required Actions
JNDI Shared Resource	<ul style="list-style-type: none">• JNDI Connection Configuration• If the resource contained an identity, an Identity Provider and an empty Keystore Provider.	<ul style="list-style-type: none">• Rename resource template if shared resource name contained a space.• Configure the Keystore Provider.

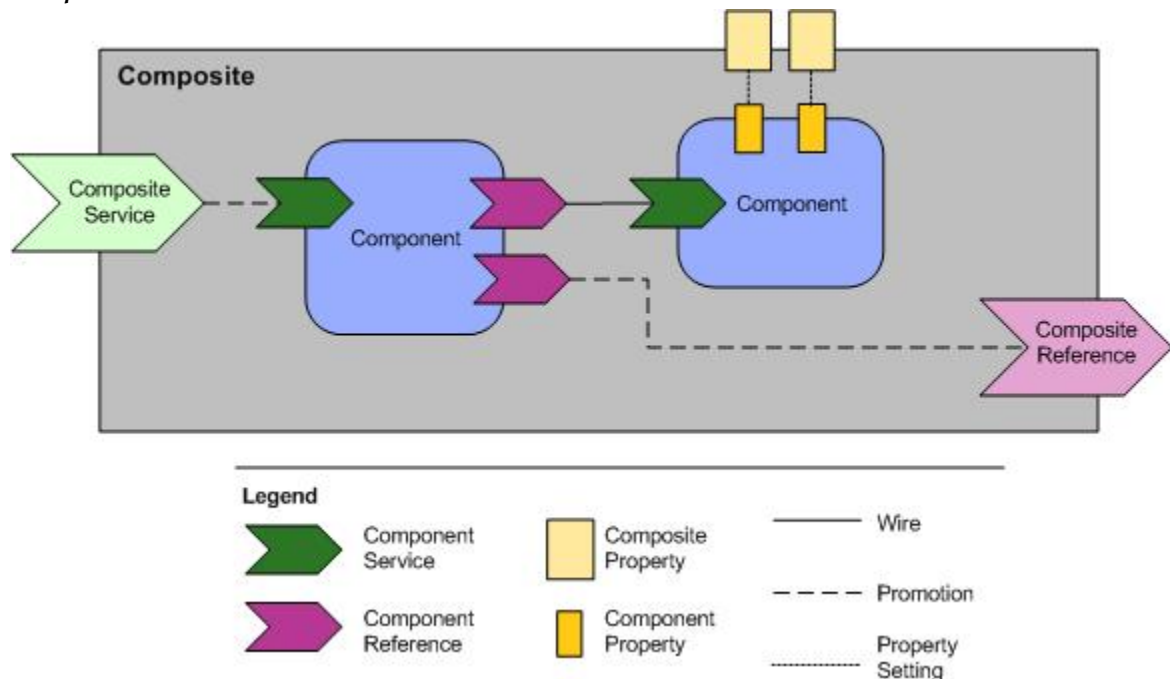
Composites

A *composite* is a configuration of services comprising an application that conforms to a service-oriented architecture. A composite contains components, services, references, the wires that interconnect them, and properties that are used to configure the components. Composites can be nested (contained by other composites). A root composite equates to an SCA application.

See the [SCA Assembly](#) specification for information on service-oriented architectures.

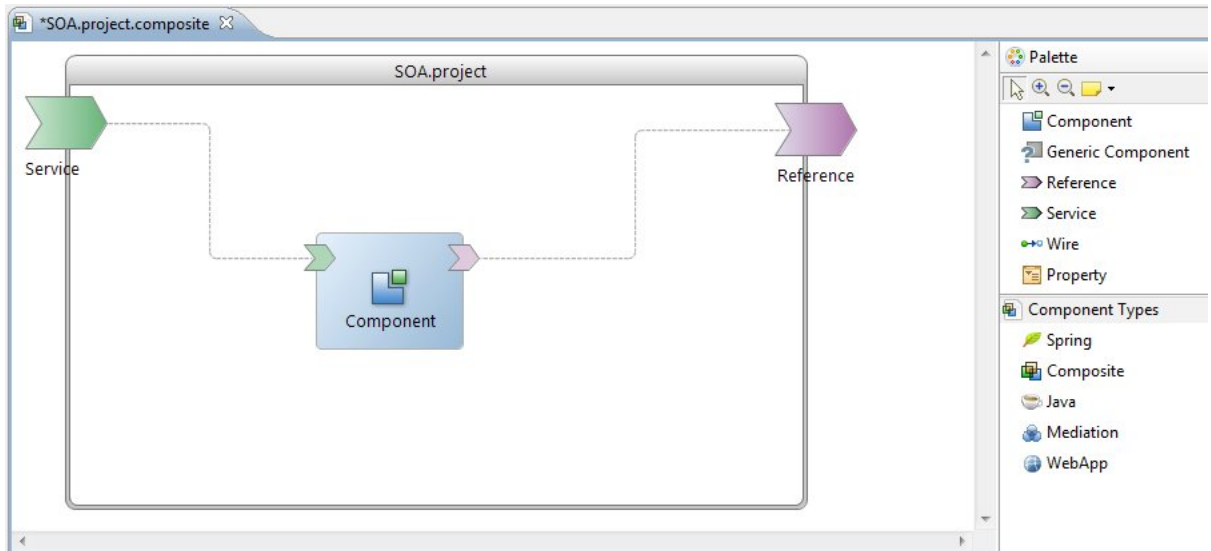
The constituents of a Composite can run in a single process on a single computer or be distributed across multiple processes on multiple computers. A complete Application might be constructed from just one Composite or it could combine several different Composites. The Components making up each Composite can be implemented using different technologies.

Composite



You edit Composites in the TIBCO Business Studio Composite Editor shown in the following screen shot.

Composite Editor



By default, composite files are stored in the `Composites` special folder in an SOA project.

Creating a Composite

You can create a composite in an existing SOA project. You can also create a composite when you create a new SOA project.

Procedure

1. Choose a starting point and follow the corresponding procedure .

Starting Point	Procedure
Composite Component	<ol style="list-style-type: none"> 1. Create a composite component. 2. Create the component implementation from the canvas or in the Problems view: <ul style="list-style-type: none"> • In the canvas, right-click the component and select Quick Fixes > Create Composite . • In the Problems view: <ol style="list-style-type: none"> 1. Right-click the error Component "Compositen" is not configured and select Quick Fix. 2. In the Quick Fix dialog, click Create Composite and click Finish.
Toolbar	<ol style="list-style-type: none"> 1. Select File > New > TIBCO SOA Resources > Composite.
Project Explorer	<ol style="list-style-type: none"> 1. Right-click a Composites folder and select New > Composite.


2. Select a folder to contain the composite.
3. In the File name field, type a name for the composite.
4. Click **Finish**.

Setting Composite Diagram Preferences

Setting composite preferences allows you to customize the user interface to work best with your project and your working style.

Procedure

1. Select **Windows > Preferences** .
The Preferences dialog displays.
2. Click **TIBCO SOA Platform > Composite Diagram** .
The composite diagram preferences display on the right. The default setting is to enable all the preferences.

Preference	Description
Show connector handles	When you hover over a service or reference, connector handles display, which when clicked allow you to wire to a matching reference or service.
Show popup bars	When you hover over a composite or composite element a pop-up bar containing elements that can be added will display.
Enable animated layout	Not used.
Enable animated zoom	When you zoom in or out, the elements in the canvas are scaled gradually.
Enable anti-aliasing	Controls the quality of the lines.
Enable connection animation	When a wire is selected, the wire displays small balls flowing from left to right. Animation can cause the CPU usage of TIBCO Business Studio to spike. You can also toggle animation by clicking the animation () button in the toolbar above the canvas.

Adding an Element to a Composite

You can add an element to a composite by using popup toolbars, the context menu, or the palette.


Popup Toolbars

TIBCO Business Studio supports a composite toolbar and a component toolbar. You can enable and disable the toolbars.

The toolbars provides the following two benefits:

- They allow you to add composite and component elements to the canvas without having to move the cursor off to the canvas to the palette.
- They enable you to hide the palette so that the canvas has increased screen area.

The composite toolbar  , which displays when the focus of the cursor is a

composite, allows you to add component to the canvas. The component toolbar  ,

which displays when the focus of the cursor is a component, allows you to add services, references, and properties to a component. The toolbars display for a few seconds and then disappear.

Enabling and Disabling Popup Toolbars

To specify whether a context-sensitive popup toolbar should display when you mouse over a composite or component:

1. Select **Window > Preferences**.
2. Click the **Diagram** node.
3. Check the **Show popup bars** checkbox.
4. Click **OK**.

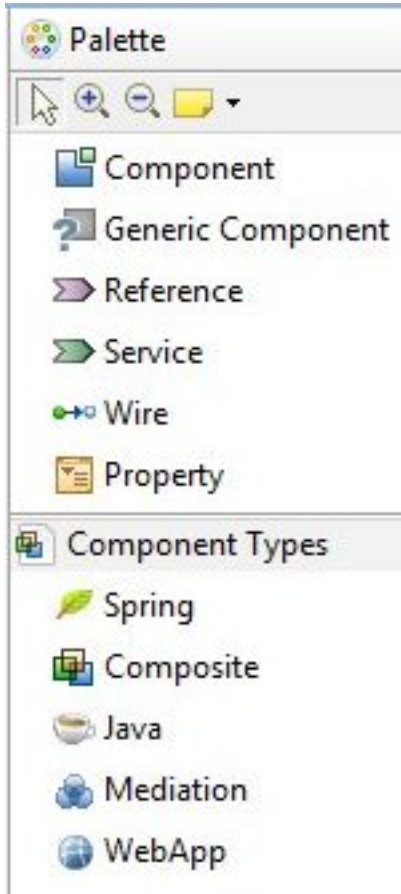
Palette

The palette allows you to select elements, add notes to the canvas, and add elements such as components, services, references, wires, and properties to the composite canvas.

The palette contains a toolbar, one unnamed section, and one named section—Component Types.

- The toolbar contains buttons to select elements within the canvas, zoom in and out, and add various types of notes to the composite canvas.
- The unnamed section contains buttons to add components, services, references, wires, and properties to the composite canvas.
- The Component Types section contains icons for adding typed components to the composite. The named section is extensible; TIBCO ActiveMatrix products can add new types of components to this section.

Palette



The palette has several features that enable you to control whether the palette is displayed and which sections within the palette are displayed. These features allow you to control the screen real estate occupied by the palette.

Toggling the Palette

The palette enables you to control whether the palette is displayed.

- To hide the palette, click the right arrow icon in the Palette heading.
- To display the palette, click the left arrow icon in the palette heading.

Selecting an Element in the Palette

To select a composite element in the palette, click the element. The element is highlighted. Once a composite element has been selected the only operations you can perform are left-click to drop the element on the canvas or click another element in the palette or toolbar.

Application Modularization

Application modularization supports reuse, and scalability, and makes it easier for different people to work on a project at the same time.

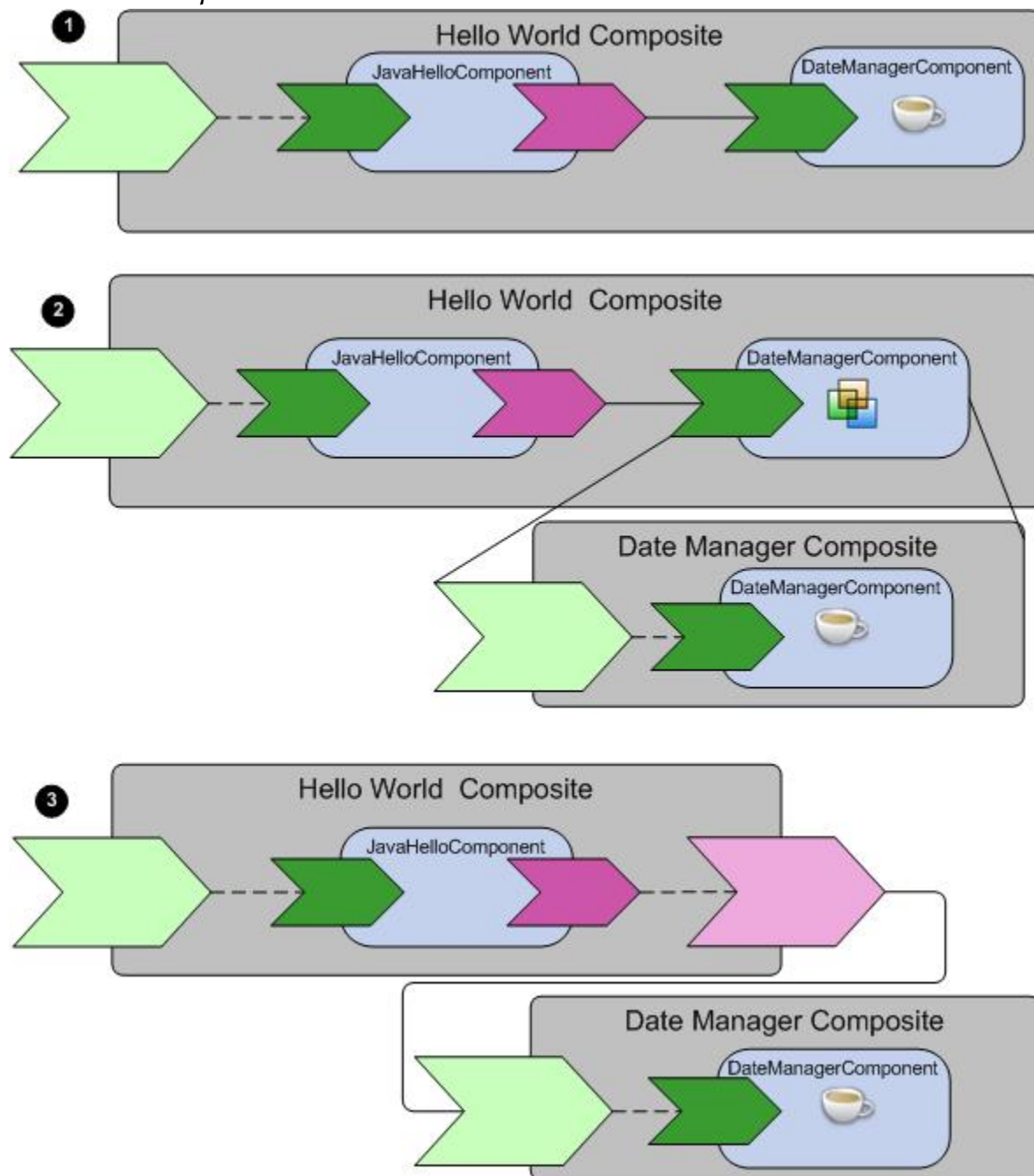
The composite is the fundamental unit of application modularization. Decomposing applications into multiple composites supports:

- Reuse - A composite can be reused as the implementation of a component in multiple composites or as a peer composite.
- Application scalability - The complexity of a given composite can be reduced by partitioning components among composites.
- Work management - Composites can be worked on independently.

There are various ways to decompose application functionality among composites. An application can consist of a root composite with internal components, a root composite with some components implemented as composites, or multiple peer composites. The following figure illustrates these three options. In the figure Hello World is a root composite. The DateManager functionality is provided in three ways:

- As an inline component
- As an inline component implemented as a nested composite
- As a peer root composite

Modularization Options



The following table summarizes the implications of the three options. The inline and nested composite approach offers reuse opportunities at design time but restrict life cycle management options. The peer composite approach offers reuse opportunities at runtime and flexible life cycle management.

Modularization Implications

	Inline (1)	Nested (2)	Peer (3)
Design Reuse	Implementation reusable	Implementation reusable	N/A

	Inline (1)	Nested (2)	Peer (3)
	Component configuration not reusable	Component configuration reusable	N/A
Runtime Reuse	N/A	N/A	Composite reusable via application and environment wiring
Life Cycle Management	Dependent	Dependent	Independent

Run Configuration Reference

For each run configuration, you can specify the applications, HTTP Connectors, Debug Node attributes, Runtime information, and Substitution Variable overrides.

Applications

The composite applications and DAAs deployed when the configuration is launched.

Advanced

Field	Description
HTTP Connectors	Ports of the HTTP connectors used by the promoted services in the composite.
Node	Debug node.
Enable Security Manager	Indicate whether a security manager is enabled for the JVM running the debug node. Default: true.
JVM Options	Options passed to the Java virtual machine running the debug node. Default: -Xmx512m -Xms128m -XX: HeapDumpOnOutOfMemoryError.
Log File	File to which node logging messages are written. Default: <i>Operating system-specific temporary directory/nodeuniqueID.log</i> .

Field	Description
Log Level	<p>Level of the logging messages written to the log.</p> <ul style="list-style-type: none"> • TRACE All events. • DEBUG Fine-grained informational events used for debugging an application. • INFO Coarse-grained informational messages that highlight the progress of the application. • WARN Potentially harmful events. • ERROR Errors that allow the application to continue running. • FATAL Errors that cause the application to fail. • OFF Blocks passing messages to a parent <p>Default: Info.</p>
Management Port	Port on which the debug node receives life cycle messages.
Node Platform Version	The platform version of the node. This value is read-only and can only be changed in the Preferences panel for RAD. See Platform Installation Reference .
OSGi Console Port	Port on which to access the node's OSGi console. Use the OSGi console to examine the runtime state of the node for debugging purposes only. For information on the features available in the OSGi console, see Explore Eclipse's OSGi console .
Runtime	Debug node runtime artifacts and properties.
Installation Directory	<p>Directory containing the debug node executable.</p> <p>Default: <i>TIBCO_HOME/tibcohost/<version></i>, which is the executable installed with TIBCO Business Studio. If you change the value, the version of the executable must be the same as the executable that is installed with TIBCO Business Studio.</p>
Instance Parent Directory	<p>Directory containing the debug node instance.</p> <p>Default: <i>workspace/metadata/.plugins/com.tibco.amf.tools.composite.launch</i>.</p>
Provisioning Wait (ms)	<p>Time to wait between provisioning requests to the debug node. If set to zero, the default, the debug launcher uses notifications from the debug node to determine when the resource instances and connectors are ready for the next deployment request. If this fails, the application could fail to deploy. In that case, set a value in this field, which will increase deployment time, because each provision request will have a wait time.</p> <p>Default: 0.</p>
Substitution Variable Overrides	Substitution variable values defined by composite applications. You cannot override substitution variable values defined by DAAs.

Common

See [Help > Help Contents > Supplemental Eclipse Help > Plug-in Development > Environment Guide > Tools > Launchers > JUnit Plug-in Test Launcher > Common Tab](#) .

Running and Debugging a Composite

TIBCO Business Studio provides an environment for executing composites in a local debug node. The debug node is a TIBCO Host instance and node combined into a single process.

You can execute composites in run mode or in debug mode. In run mode, the composite executes, but you cannot suspend the execution or examine the project. In debug mode, you can suspend and resume execution, inspect and modify variables, and evaluate expressions. Running or debugging in a local node is also referred to as rapid application development (RAD) because you do not have to package and deploy the composite in order to test it.

To run or debug a composite, you create a run or debug configuration in which you specify: one or more composites, distributed application archives containing libraries, and resource and debug node properties. When you run the configuration, TIBCO Business Studio:

- Creates and starts the node and installs resources required by the composite
- Packages the composite into a distributed application archive and uploads it to the node
- Creates, deploys, and starts the application

If you change a composite and rerun the configuration, the entire node is reset and all application fragments are redeployed.

In TIBCO Business Studio, when a project is launched in debug and run mode, the console message indicates the port number configured for the HTTP connector. For example:

```
*** HTTP connector listening on port '5002' ***
```

```
Use URL: http://localhost:5002/sample/ for SOAP or Web access.
```

Limitations

The following features that are available when you run a composite on a remote node are not supported when running composites on a local debug node:

- Wiring between components running on different nodes.
- Successive deployments on the same node. As a consequence, applications that behave differently depending on component availability cannot be tested.
- A debug node cannot start a JMS server. If the composite has JMS resource templates that require a connection to a JMS server you must manually start the server before running the composite.
- Policy sets and WS-Addressing, transactions, and security features that rely on policies.
- Wiring using a Service Virtualization binding between different composites deployed on the same node. Use a SOAP binding instead.
- One node is supported per configuration. However, you can run different configurations on different nodes as long as ports and other system resources don't conflict.

Creating a Run Configuration

Before you can run a composite, you create a run configuration. After you save the configuration, it becomes available from the Run menu.

Procedure

1. Select **Run > Run Configurations**.
The Run Configurations dialog displays.
2. Double-click **Composite Application**.
A new run configuration named New_configuration is created and opened in the configuration editor.
3. In the Name field, type a configuration name.
4. In the Composite Applications area, click **Add**, select a composite or DAA, and click **OK**.
The composite or DAA is added to the Composite Applications list.
5. If you have installed third-party drivers using TIBCO Configuration Tool and if your application requires those drivers on the debug node, add driver features, as follows.
 - a) Click **Add Driver Features**.
The Installed Driver Features dialog displays with a list of installed drivers.
 - b) Click the feature you want and click **OK**.
The feature is added to the Composite Applications list.
6. Click the **Advanced** tab to optionally update run configuration properties as described in [Run Configuration Reference](#).
 - a) Expand a node.
 - b) Click the Value column, type a new value, and press **Enter**.
 - c) Click **Apply**.
The configuration is saved.
7. Click the **Common** tab and check the **Run** checkbox in the Display in favorites menu
8. Click **Apply** and click **Close**.
The configuration is saved and added to the Run menu options.

Running a Composite

After you create a composite configuration, you can run a composite and explore its state in the Administrator Explorer tab. You might have to allow the TIBCO Java Launcher if your Windows firewall blocks it.

Prerequisites

Create a Run Configuration. See [Creating a Run Configuration](#).

Procedure

1. In the toolbar, click the run icon () and select a run configuration.
The debug node is created and the node process is started.

2. On Windows systems, click **Unblock** if the following dialog pops up:



The node process startup completes, and the system deploys the composite to the debug node and starts it.

3. In the dialog that asks whether to show the Administrator Explorer View, click **Yes**.
4. Click the **Administrator Explorer** tab.
You can now examine the state of the composites in the run configuration and its constituent connectors, components, resource instances, and endpoints.

Run Configuration Reference

For each run configuration, you can specify the applications, HTTP Connectors, Debug Node attributes, Runtime information, and Substitution Variable overrides.

Applications

The composite applications and DAAs deployed when the configuration is launched.

Advanced

Field	Description
HTTP Connectors	Ports of the HTTP connectors used by the promoted services in the composite.
Node	Debug node.
Enable Security Manager	Indicate whether a security manager is enabled for the JVM running the debug node. Default: true.
JVM Options	Options passed to the Java virtual machine running the debug node. Default: -Xmx512m -Xms128m -XX: HeapDumpOnOutOfMemoryError.

Field	Description
Log File	File to which node logging messages are written. Default: <i>Operating system-specific temporary directory/nodeuniqueID.log</i> .
Log Level	Level of the logging messages written to the log. <ul style="list-style-type: none"> • TRACE All events. • DEBUG Fine-grained informational events used for debugging an application. • INFO Coarse-grained informational messages that highlight the progress of the application. • WARN Potentially harmful events. • ERROR Errors that allow the application to continue running. • FATAL Errors that cause the application to fail. • OFF Blocks passing messages to a parent Default: Info.
Management Port	Port on which the debug node receives life cycle messages.
Node Platform Version	The platform version of the node. This value is read-only and can only be changed in the Preferences panel for RAD. See Platform Installation Reference .
OSGi Console Port	Port on which to access the node's OSGi console. Use the OSGi console to examine the runtime state of the node for debugging purposes only. For information on the features available in the OSGi console, see Explore Eclipse's OSGi console .
Runtime	Debug node runtime artifacts and properties.
Installation Directory	Directory containing the debug node executable. Default: <i>TIBCO_HOME/tibcohost/<version></i> , which is the executable installed with TIBCO Business Studio. If you change the value, the version of the executable must be the same as the executable that is installed with TIBCO Business Studio.
Instance Parent Directory	Directory containing the debug node instance. Default: <i>workspace/metadata/.plugins/com.tibco.amf.tools.composite.launch</i> .
Provisioning Wait (ms)	Time to wait between provisioning requests to the debug node. If set to zero, the default, the debug launcher uses notifications from the debug node to determine when the resource instances and connectors are ready for the next deployment request. If this fails, the application could fail to deploy. In that case, set a value in this field, which will increase deployment time, because each provision request will have a wait time. Default: 0.

Field	Description
Substitution Variable Overrides	Substitution variable values defined by composite applications. You cannot override substitution variable values defined by DAAs.

Common

See **Help > Help Contents > Supplemental Eclipse Help > Plug-in Development > Environment Guide > Tools > Launchers > JUnit Plug-in Test Launcher > Common Tab** .

Creating a Debug Configuration

Before you can debug a composite, you have to create a debug configuration.

Procedure

1. Select **Run > Debug Configurations**.
2. Double-click **Composite Application** and configure the composite.
 - a) In the Name field, type a configuration name.
 - b) In the Composite Applications area, click **Add** to add a composite file or **Add DAA** to add a DAA.
 - c) Select a composite or DAA and click **OK**.
3. Click the **Debug Participants** tab.
4. Uncheck the checkboxes next to the debuggers you do not want to run when the configuration is launched.
5. Optionally, click the **Advanced** tab and update [debug session properties](#).
 - a) Expand a node.
 - b) Click the Value column, type a new value, and press **Enter**.
 - c) Click **Apply**.
The configuration is saved.
6. Click the **Common** tab and check the **Debug** checkbox in the Display in favorites menu.
7. Click **Apply** and click **Close**.

Result

The configuration is saved and added to the Debug menu options.


Debugging a Composite

After you have created a debug configuration, you can debug a composite. You might have to allow the TIBCO Java Launcher if your Windows firewall blocks it.

Prerequisites

Create a Debug Configuration. See [debug configuration](#).

Procedure

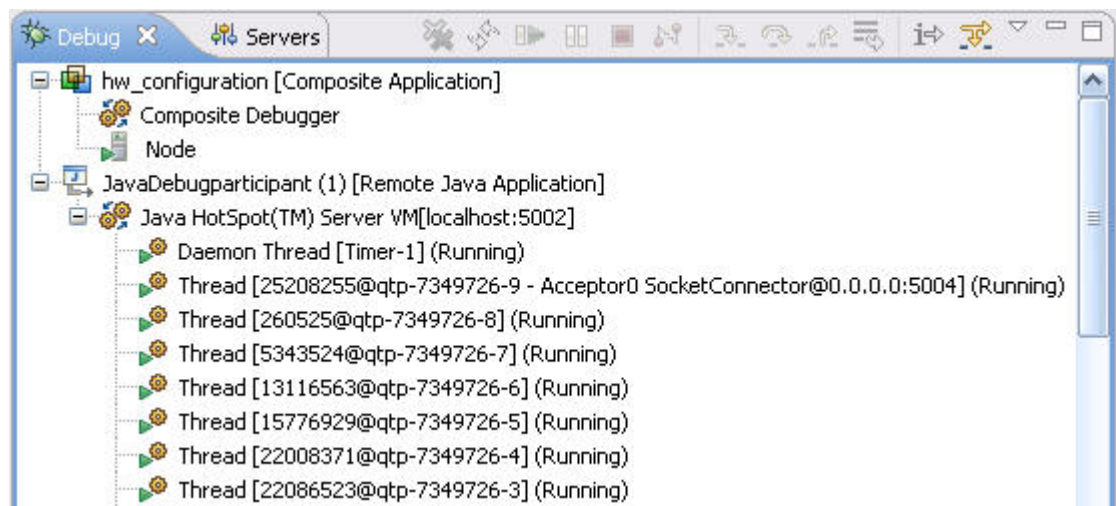
1. In the toolbar, click the debug icon () and select the debug configuration.
The debug node is created and the node process is started.

2. On Windows systems, click **Unblock** if you see the following dialog pop up:



The node process startup completes, the composite is deployed to the debug node, and is started. A dialog pops up asking whether to show the Composite Applications View.

3. Click **Yes**.
The Composite Applications View displays.
4. Select **Window > Open Perspective > Other...**
The Open Perspective dialog displays.
5. Click **Debug** and click **OK**.
The Debug perspective displays. The Debug view displays the participants you selected when you created the debug configuration.



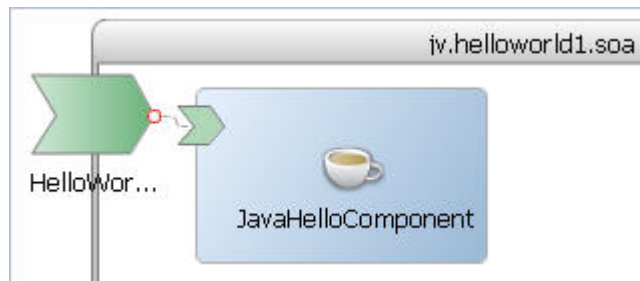
Setting Breakpoints

Sets a breakpoint for the request and response.

Procedure

- Right-click a composite or component service or reference and select **Set Breakpoint**.

A red circle breakpoint badge is added to the selected element.



Removing Breakpoints

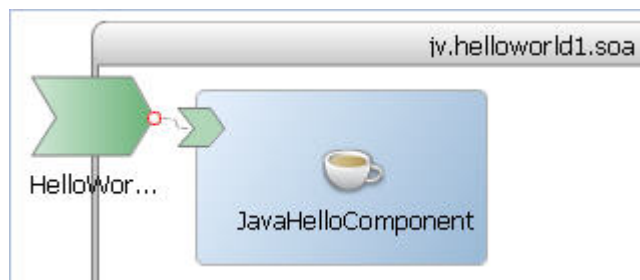
Procedure

- Right-click a composite element and select **Remove Breakpoint**.

Configuring Breakpoints

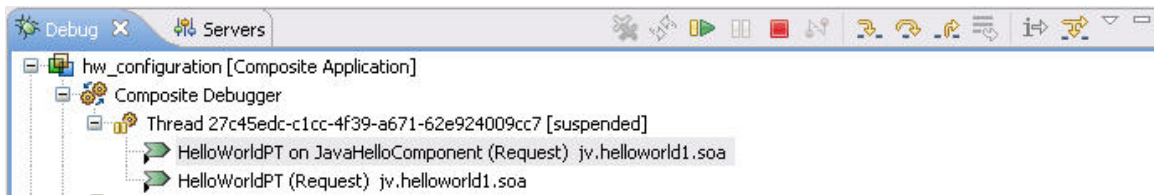
Procedure

1. Right-click the canvas and select **Configure Breakpoints**.
The Configure Breakpoints dialog displays.
2. Check the checkboxes next to the composite or composite element on which to set the breakpoint.
The Select message filter on breakpoint checkbox is enabled.
3. To choose a message type, check the **Select message filter on breakpoint** checkbox.
The On Request and On Response checkboxes are enabled.
4. Check a message checkbox.
The breakpoint is enabled for the checked message.
5. Click **OK**.
A red circle breakpoint badge is added to the selected element.

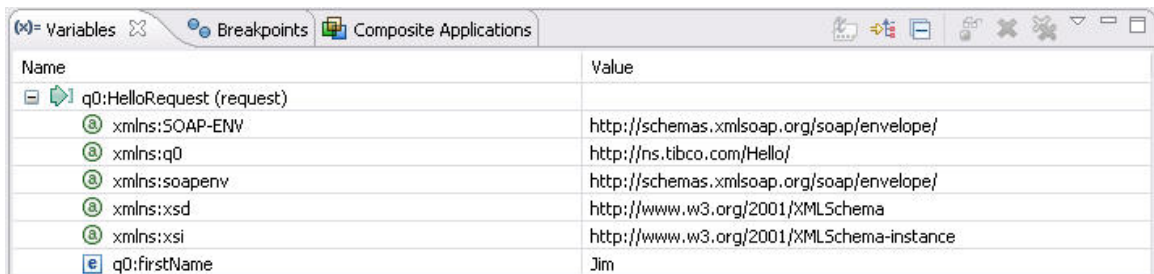


Controlling Composite Execution

When an executing composite reaches a breakpoint, execution is suspended and the Debug view is updated with the current debugger state.



To display the state of a suspended object, click the object in the Debug view. The state displays in the Variables view:



To update the value of a variable, click the Value column and type the new value.

Click buttons in the Debug view toolbar to resume or terminate execution or step into, over, or out of an object:



Debug Configuration Reference

Applications

The composite applications and DAAs deployed when the configuration is launched.

Debug Participants

The remote debuggers to run when the configuration is launched:

- Composite Debugger - Run a composite debugger.
- Java Component Debugger - Run a Java debugger. For information on how to use this debugger, see **Supplemental Eclipse Help > Java Development User Guide > Tasks > > Running and Debugging**.

Advanced

Field	Description
Debug Participants	
Composite Debugger - Port	The port of the composite debugger.

Field	Description
XXX Component Debugger - Port	The port of the XXX component debugger, where XXX is Java or Spring.

Field	Description
HTTP Connectors	Ports of the HTTP connectors used by the promoted services in the composite.
Node	Debug node.
Enable Security Manager	Indicate whether a security manager is enabled for the JVM running the debug node. Default: true.
JVM Options	Options passed to the Java virtual machine running the debug node. Default: -Xmx512m -Xms128m -XX:+HeapDumpOnOutOfMemoryError.
Log File	File to which node logging messages are written. Default: <i>Operating system-specific temporary directory/nodeuniqueID.log</i> .
Log Level	Level of the logging messages written to the log. <ul style="list-style-type: none"> • TRACE All events. • DEBUG Fine-grained informational events used for debugging an application. • INFO Coarse-grained informational messages that highlight the progress of the application. • WARN Potentially harmful events. • ERROR Errors that allow the application to continue running. • FATAL Errors that cause the application to fail. • OFF Blocks passing messages to a parent Default: Info.
Management Port	Port on which the debug node receives life cycle messages.
OSGi Console Port	Port on which to access the node's OSGi console. Use the OSGi console to examine the runtime state of the node for debugging purposes only. For information on the features available in the OSGi console, see Explore Eclipse's OSGi console .
Runtime	Debug node runtime artifacts and properties.

Field	Description
Installation Directory	Directory containing the debug node executable. Default: <i>TiBCO_HOME/tibcohost/3.2</i> , which is the executable installed with TIBCO Business Studio. If you change the value, the version of the executable must be the same as the executable that is installed with TIBCO Business Studio.
Instance Parent Directory	Directory containing the debug node instance. Default: <i>workspace/metadata/.plugins/com.tibco.amf.tools.composite.launch.</i>
Provisioning Wait (ms)	Time to wait between provisioning requests to the debug node. If set to zero, the default, the debug launcher uses notifications from the debug node to determine when the resource instances and connectors are ready for the next deployment request. If this fails, the application could fail to deploy. In that case, set a value in this field, which will increase deployment time, because each provision request will have a wait time. Default: 0.
Substitution Variable Overrides	Substitution variable values defined by composite applications. You cannot override substitution variable values defined by DAAs.

Common

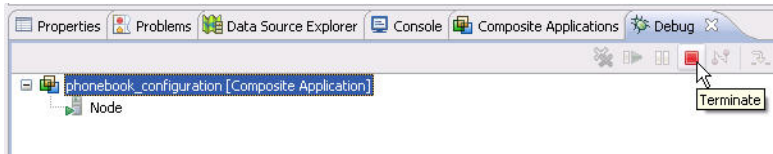
See **Help > Help Contents > Supplemental Eclipse Help > Plug-in Development > Environment Guide > Tools > Launchers > JUnit Plug-in Test Launcher > Common Tab** .

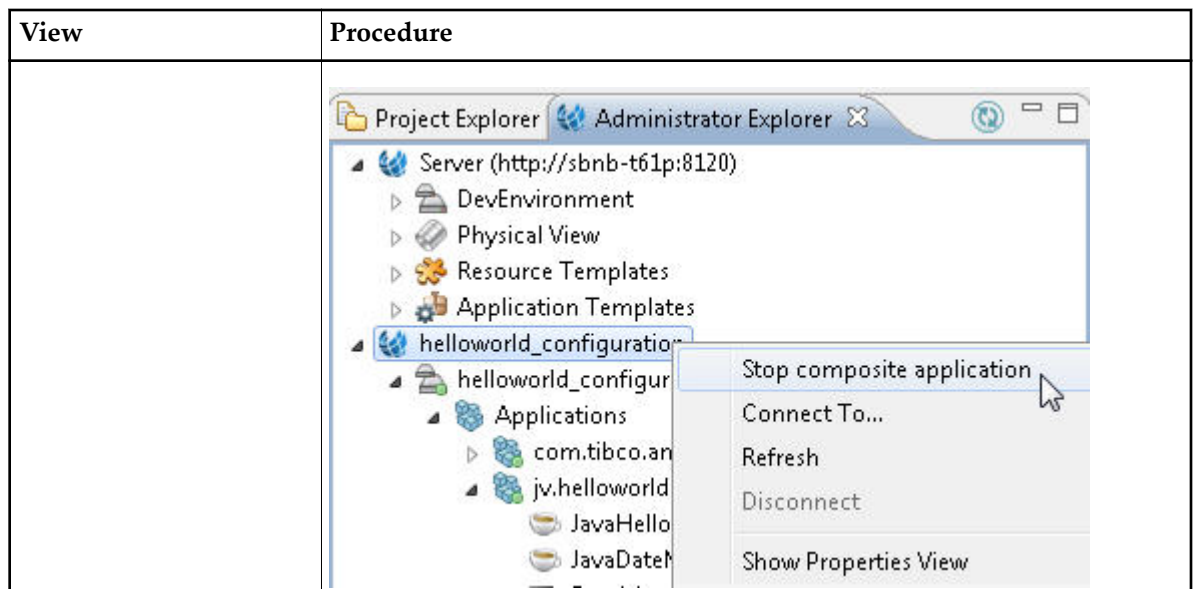
Terminating a Configuration

You can terminate a configuration from the Debug menu or from Administrator Explorer.

Procedure

1. Choose a view and follow the appropriate procedure.

View	Procedure
Debug	<ol style="list-style-type: none"> 1. Select Window > Show View > Other 2. Select Debug > Debug . 3. Click the configuration and click Terminate. 
Administrator Explorer	<ol style="list-style-type: none"> 1. Right-click a configuration and select Stop composite application.



- The application is stopped and undeployed and the debug node is stopped. In the Debug View, if you click the Terminate button a second time, the debug node is terminated immediately.

Composite Reference

General Tab

Field	Description
Name	The name of the composite.
Version	The version of the composite. Default: 1.0.0.qualifier.
Namespace	The namespace of the composite.
Description	Optional composite description.

Properties Tab

Field	Description
Name	Name of the property. Default: Property1.
Type	Type of the property: String, Boolean, Double, Integer, or a resource template. Default: String.
Source	Source of the property value. For a component property, a literal or composite property. For a composite, resource template, or binding property, a literal or a substitution variable. Default: None.

Field	Description
Value	Property value. Default: None.
Required	The property value must be set before packaging the composite containing the property into a DAA. Default: Unchecked.

Services Tab

WSDL Interface

Field	Description
Name	The name of the service. Default: Servicen, where <i>n</i> is an integer.
Port Type	The port type of the service.
WSDL Location	The location of the WSDL file that defines the port type. Read-only.

Context Parameters

Column	Description
Name	The name of the context parameter.
Operations	The operations in the port type of the service or reference to which the parameter applies.
Direction	The direction of the message to which the parameter applies: Input, Output, Fault. If Fault, the list of fault messages.
Type	The type of the parameter: <ul style="list-style-type: none"> • Basic - a basic property. • Message - a message associated with the binding. The binding cannot reference a message used by an operation in the port type. • Bag - a collection of properties.
Definition	<ul style="list-style-type: none"> • Basic - the definition of the property. If the Direction is Input or Output, one of string, int, long, or boolean. If the Direction is Fault, one of string, int, long, boolean, QName, or URI. • Message - the definition of the message, consisting of a message and a message part.

Advanced

Field	Description
Conversational	<p>Indicate that there is a sequence of operations in the port type that must be called in order to achieve some higher level goal. Supported only when WS-Reliable Messaging is enabled for a SOAP binding.</p> <p>Default: False.</p>

References Tab*WSDL Interface*

Field	Description
Name	<p>The name of the service.</p> <p>Default: <i>Servicen</i>, where n is an integer.</p>
Port Type	The port type of the service.
WSDL Location	The location of the WSDL file that defines the port type. Read-only.

Context Parameters

Column	Description
Name	The name of the context parameter.
Operations	The operations in the port type of the service or reference to which the parameter applies.
Direction	The direction of the message to which the parameter applies: Input, Output, Fault. If Fault, the list of fault messages.
Type	<p>The type of the parameter:</p> <ul style="list-style-type: none"> Basic - a basic property. Message - a message associated with the binding. The binding cannot reference a message used by an operation in the port type. Bag - a collection of properties.
Definition	<ul style="list-style-type: none"> Basic - the definition of the property. If the Direction is Input or Output, one of string, int, long, or boolean. If the Direction is Fault, one of string, int, long, boolean, QName, or URI. Message - the definition of the message, consisting of a message and a message part.

Advanced

Field	Description
Wired by Implementation	Indicate that the implementation will dynamically wire the reference to a service when the component runs. Default: False.
Conversational	Indicate that there is a sequence of operations in the port type that must be called in order to achieve some higher level goal. Supported only when WS-Reliable Messaging is enabled for a SOAP binding. Default: False.
End Operation	The operation that signifies the end of the conversation. Enabled when Conversational is checked.

Policies Tab[Policy Management](#)**Rulers & Grid Tab**

Field	Description
Show Ruler	Displays the ruler.
Show Grid	Displays the grid.
Grid In Front	Displays the grid in front of the Composite window.
Ruler Units	Sets the ruler units to inches, centimeters, or pixels.
Grid Spacing	Specifies the grid spacing in the units specified.
Color	Specifies the color of the grid lines.
Style	Specifies the style of the grid line.
Snap To Grid	Automatically aligns the item to the grid points when you move the item.
Snap to Shapes	Aligns the item to the nearest shape.
Restore Defaults	Enables you to restore the default values.

Appearance Tab

Field	Description
Fonts and Colors	Specifies the font and color.
Lines and Arrows	Specifies the style of lines and arrows.

Components

A *Component* is the basic element of business function. It is defined at design time.

Components are configured instances of implementations. More than one Component can use and configure the same implementation.

Components can have Services, References, and properties. All of these can be promoted to the Composite level during design time. Promotion enables an ActiveMatrix Administrator to wire or configure Services, References, and properties when the Application is deployed. Services, References, and properties that are not promoted are private to the Application and are set at design time only.

Components can have several different types of dependencies. Components can express dependencies on product features, custom features, other Components, and resources. All of a Component's dependencies must be satisfied for it to be deployed to a Node.


Components can be deployed to multiple Nodes for fault tolerance or load balancing.

Creating a Component

You can create a component by using the SOA Project wizard or manually.

Procedure

- Choose an option and follow the relevant procedure.

Option	Procedure
SOA Project Wizard	<ol style="list-style-type: none"> Create an SOA project. <ul style="list-style-type: none"> SOA Project from Implementation - Creates a component of the type defined by the implementation. SOA Project from WSDL - Creates a component of the type selected in the wizard. Basic SOA Project - Creates an abstract component.
Manual	<ol style="list-style-type: none"> Choose an option: <ul style="list-style-type: none"> Hover over a composite canvas and click an icon in the pop-up. Click the component icon  in the Palette and click the composite canvas. Adds an abstract component named <i>Componentn</i> to the canvas. Click an icon in the Component Types section of the Palette and click the canvas. Adds a component type named <i>ComponentType</i> to the canvas. Right-click the canvas and select Add > Component or Add > ComponentType Accept the default name or type a name in the activated input area and press Enter.

All non-abstract components created via the manual method will display error badges indicating that the component implementation is not configured. Refer to the documentation for the component type for information on how to configure the implementation.

Changing the Implementation Type of a Component

You can change a component's implementation type from the Properties view.


Procedure

1. Click a component.
2. In the Properties view, click the **General** tab.
3. In the Type field, click the **Change** link.
The Change Implementation Type dialog displays.
4. Click an implementation type and click **Finish**.

Configuring a Components External Custom Feature

If your component implementation references a library, you must add the custom feature that contains the library plug-in to the component's configuration.

Procedure

1. Click the component.
2. In the Properties view, click the **Implementation** tab.
3. Click the  button to the right of the Features table.
The Select a Feature dialog displays.
4. Click the feature containing the shared library plug-in.
5. Click **OK**.
The feature is added to the component's Features list.

Result

The following screenshot shows a Phonebook component's Implementation tab with the component implementation custom feature and a database driver library custom feature.

Class: Browse...

Location:


Package Version:

Minimum: Inclusive ▼

Maximum: Exclusive ▼

Features:

Feature ID	Version Range
jv.phonebook.soa_jv.phonebook.soa	[1.0.0,2.0.0)
jv.hsqldb.jv.feature	[1.9.0.006,2.0.0)
...	

+ × 

Component Reference

General Tab

Field	Description
Name	The name of the component.
Type	The implementation type of the component.
Description	Optional component description.
Version	The version of the component. Default: 1.0.0.qualifier.

Implementation Tab

Implementation-type specific. Refer to the implementation type documentation.

Properties Tab

Field	Description
Name	Name of the property. Default: Property1.
Type	Type of the property: String, Boolean, Double, Integer, or a resource template. Default: String.
Source	Source of the property value. For a component property, a literal or composite property. For a composite, resource template, or binding property, a literal or a substitution variable. Default: None.
Value	Property value. Default: None.
Required	The property value must be set before packaging the composite containing the property into a DAA. Default: Unchecked.

Services Tab

WSDL Interface

Field	Description
Name	The name of the service. Default: Servicen, where <i>n</i> is an integer.
Port Type	The port type of the service.
WSDL Location	The location of the WSDL file that defines the port type. Read-only.

Context Parameters

Column	Description
Name	The name of the context parameter.
Operations	The operations in the port type of the service or reference to which the parameter applies.
Direction	The direction of the message to which the parameter applies: Input, Output, Fault. If Fault, the list of fault messages.
Type	The type of the parameter: <ul style="list-style-type: none"> • Basic - a basic property. • Message - a message associated with the binding. The binding cannot reference a message used by an operation in the port type. • Bag - a collection of properties.
Definition	<ul style="list-style-type: none"> • Basic - the definition of the property. If the Direction is Input or Output, one of string, int, long, or boolean. If the Direction is Fault, one of string, int, long, boolean, QName, or URI. • Message - the definition of the message, consisting of a message and a message part.

Advanced

Field	Description
Conversational	Indicate that there is a sequence of operations in the port type that must be called in order to achieve some higher level goal. Supported only when WS-Reliable Messaging is enabled for a SOAP binding. Default: False.

References Tab

WSDL Interface

Field	Description
Name	The name of the service. Default: Servicen, where <i>n</i> is an integer.
Port Type	The port type of the service.
WSDL Location	The location of the WSDL file that defines the port type. Read-only.

Context Parameters

Column	Description
Name	The name of the context parameter.
Operations	The operations in the port type of the service or reference to which the parameter applies.
Direction	The direction of the message to which the parameter applies: Input, Output, Fault. If Fault, the list of fault messages.
Type	The type of the parameter: <ul style="list-style-type: none"> • Basic - a basic property. • Message - a message associated with the binding. The binding cannot reference a message used by an operation in the port type. • Bag - a collection of properties.
Definition	<ul style="list-style-type: none"> • Basic - the definition of the property. If the Direction is Input or Output, one of string, int, long, or boolean. If the Direction is Fault, one of string, int, long, boolean, QName, or URI. • Message - the definition of the message, consisting of a message and a message part.

Advanced

Field	Description
Wired by Implementation	Indicate that the implementation will dynamically wire the reference to a service when the component runs. Default: False.

Field	Description
Conversational	Indicate that there is a sequence of operations in the port type that must be called in order to achieve some higher level goal. Supported only when WS-Reliable Messaging is enabled for a SOAP binding. Default: False.
End Operation	The operation that signifies the end of the conversation. Enabled when Conversational is checked.

Policies Tab

[Policy Management](#)

Appearance Tab

Field	Description
Custom image location	Specifies the location of any custom image to be used for the icon.
Fonts and Colors	Specifies the font and color.
Lines and Arrows	Specifies the style of lines and arrows.
Gradient	Specifies the color gradient.

Services and References

Applications interact via services and references. A *service* is a set of operations and the messages required by the operations. A *reference* identifies the service consumed by a component or composite. Applications offer services and invoke references to other services. An application's services and references are promoted from the services and references of the components it contains.

Component services can be consumed by other components within the composite or promoted as composite services for use by consumers outside the composite. A composite service has an interface and one or more bindings.

Component references consume services provided by other components in the same composite or services provided outside the composite. A composite reference has an interface and one binding.

TIBCO ActiveMatrix services are described using WSDL port types. TIBCO ActiveMatrix supports services that comply with the [WSDL 1.1 specification](#).

Interfaces







An *Interface* defines the contract for Services and References. Services and References can interact only when they have the same Interface. An Interface defines one or more operations and each operation has zero or one Request (input) message and zero or one Response (output) message. The Request and Response messages may be simple types such as strings and integers or they may be complex types. In the current release, TIBCO ActiveMatrix supports WSDL 1.1 port type Interfaces.

Adding a Service or Reference to a Component

You can add a service or reference to a component from the component, the palette, or the Properties view.

Procedure

- Choose a starting point and follow the appropriate procedure:

Starting Point	Procedure
Component	<ul style="list-style-type: none"> 1. Click the component. 2. Click the service  or reference  icon in the component pop-up toolbar. 1. Right-click the component. 2. Select  Add > Service or Reference.
Properties View	<ol style="list-style-type: none"> Click the component. In the Properties view, click the Services or References tab. Click .
Palette	<ol style="list-style-type: none"> Click the Service  or Reference  icon in the Palette. The selected Palette entry is highlighted. Click the component.






A component service or reference icon is added to the component. An error badge is added to the service or reference icon and the component. The following errors are added to the Problems view:

- WSDL Interface in the "*ServiceName*" or "*ReferenceName*" is not configured.
- Reference - The required component reference "*ReferenceName*" is not wired to any component service or promoted reference .
- Non-abstract components - The component *ComponentName* is out of sync with its implementation.

Adding a Service or Reference to a Composite

You can add a service or reference to a composite from the canvas, the palette, or the Properties view.

Procedure

Starting Point	Procedure
Canvas	<ul style="list-style-type: none"> • 1. Right-click and select Add > Promoted Service or Reference. • 1. Hover in the composite canvas near the composite name. 2. Click the Service  or Reference  icon in the popup toolbar.
Palette	<ol style="list-style-type: none"> 1. Click the Service  or Reference  icon in the Palette. The selected Palette entry is highlighted. 2. <ul style="list-style-type: none"> • Service - Hover over the left edge of the composite. • Reference - Hover over the left edge of the composite. 3. When the edge is highlighted, click the left mouse button.
Properties View	<ol style="list-style-type: none"> 1. Click the Services or References tab. 2. Click the Add icon ( Add).

A service or reference with an error badge named *ServiceN* or *ReferenceN* respectively is added to the composite.

Promoting a Component Service or Reference

You can promote an individual component service or reference, or promote all at the same time.

Procedure

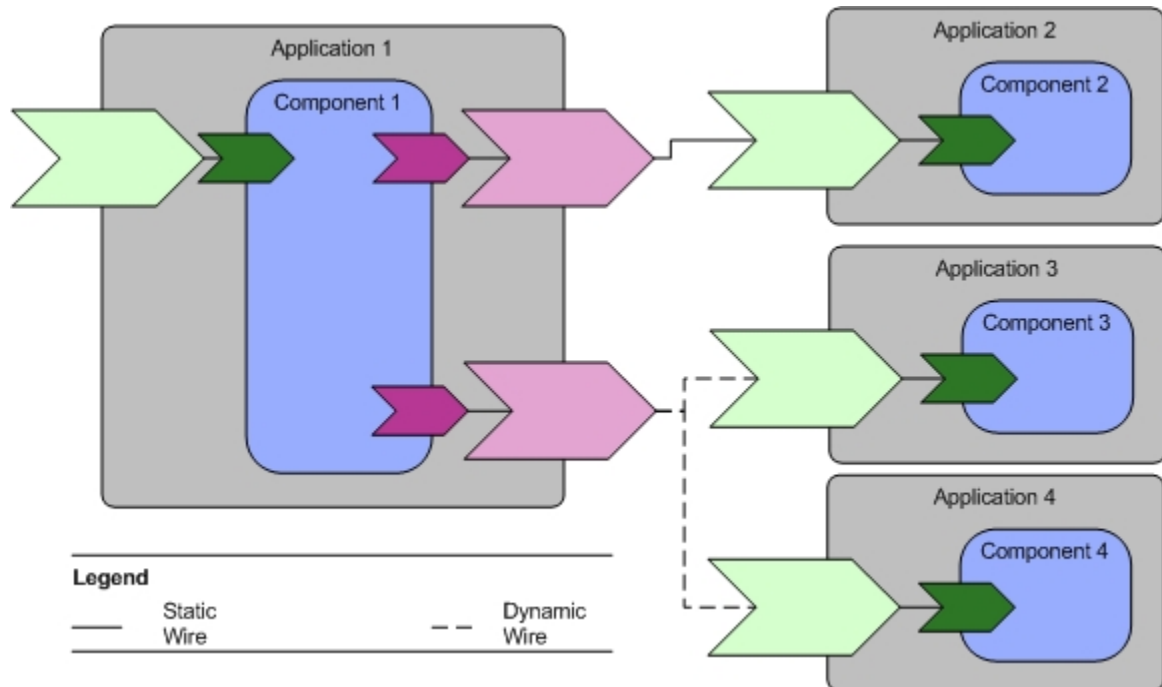
- - Right-click the component service or reference and select **Promote**.
 - Right-click the component and select **Promote All** .

Static and Dynamic Wiring

The connection between a service and a reference is called a wire. A wire can be static or dynamic

The following illustration shows the two types of wiring.

Static and Dynamic Wiring



A *static wire* is explicitly specified in a composite. A reference can be statically wired to at most one service. A *dynamic wire* is created by the component invoking the reference. A reference can be dynamically wired to many services. Dynamic wiring is useful in circumstances where the target service of a reference is not known until runtime. For example, a target service could be invoked based on the zip code from which a request originates.


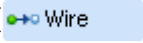
Dynamic wiring is supported by Virtualization and SOAP (SOAP/HTTP and SOAP/JMS) bindings and Mediation, Java, Spring, and WebApp components. When using dynamic wiring, both the component and promoted reference must be configured to use dynamic wiring. The target service is identified by the component implementation that invokes the reference. Depending on the component type, the component implementation can identify the target service either by providing the application and service name or by providing the URI of the service. For further information, see the specific component development manuals.

Creating a Static Wire Between a Service and a Reference

You can wire a service to a reference by selecting the service, the reference, or by using the wire icon on the palette.

Procedure

1. Choose a starting point and follow the appropriate procedure.

Starting Point	Procedure
Service or reference	<ol style="list-style-type: none"> 1. Hover over the service or reference. The connection handles icon () displays. 2. Press and hold the left mouse button.
Palette	<ol style="list-style-type: none"> 1. Click the wire () icon. 2. Move the cursor to a service or reference.

Starting Point	Procedure
	3. When the cursor changes shape, press and hold the left mouse button.

2. Draw a wire to a service or reference with a matching WSDL interface.
3. Release the mouse button.

Configuring a Reference for Dynamic Wiring

The composite reference can only have a Virtualization or SOAP binding.

Procedure

1. Click a component reference.
2. In the General tab of the Properties view, expand the **Advanced** section.
3. Check the **Wired by Implementation** checkbox.
4. Click the composite reference promoted from the component reference.
5. In the General tab of the Properties view, expand the Advanced section.
6. Check the Wired by Implementation checkbox.
7. Save the composite.

Service Reference

General Tab

WSDL Interface

Field	Description
Name	The name of the service. Default: Servicen, where <i>n</i> is an integer.
Port Type	The port type of the service.
WSDL Location	The location of the WSDL file that defines the port type. Read-only.

Context Parameters

Column	Description
Name	The name of the context parameter.
Operations	The operations in the port type of the service or reference to which the parameter applies.
Direction	The direction of the message to which the parameter applies: Input, Output, Fault. If Fault, the list of fault messages.

Column	Description
Type	<p>The type of the parameter:</p> <ul style="list-style-type: none"> • Basic - a basic property. • Message - a message associated with the binding. The binding cannot reference a message used by an operation in the port type. • Bag - a collection of properties.
Definition	<ul style="list-style-type: none"> • Basic - the definition of the property. If the Direction is Input or Output, one of string, int, long, or boolean. If the Direction is Fault, one of string, int, long, boolean, QName, or URI. • Message - the definition of the message, consisting of a message and a message part.

Advanced

Field	Description
Conversational	<p>Indicate that there is a sequence of operations in the port type that must be called in order to achieve some higher level goal. Supported only when WS-Reliable Messaging is enabled for a SOAP binding.</p> <p>Default: False.</p>

Bindings Tab

See [SOAP Binding Reference](#) and [JMS Binding Reference](#).

Policies Tab

[Policy Management](#).

Appearance Tab

Field	Description
Fonts and Colors	Specifies the font and color.
Lines and Arrows	Specifies the style of lines and arrows.
Gradient	Specifies the color gradient.

Reference Reference

General Tab

WSDL Interface

Field	Description
Name	The name of the reference. Default: Referencen, where <i>n</i> is an integer.
Port Type	The port type of the reference.
WSDL Location	The location of the WSDL file that defines the port type. Read-only.

Context Parameters

Column	Description
Name	The name of the context parameter.
Operations	The operations in the port type of the service or reference to which the parameter applies.
Direction	The direction of the message to which the parameter applies: Input, Output, Fault. If Fault, the list of fault messages.
Type	The type of the parameter: <ul style="list-style-type: none"> • Basic - a basic property. • Message - a message associated with the binding. The binding cannot reference a message used by an operation in the port type. • Bag - a collection of properties.
Definition	<ul style="list-style-type: none"> • Basic - the definition of the property. If the Direction is Input or Output, one of string, int, long, or boolean. If the Direction is Fault, one of string, int, long, boolean, QName, or URI. • Message - the definition of the message, consisting of a message and a message part.

Advanced

Field	Description
Wired by Implementation	Indicate that the implementation will dynamically wire the reference to a service when the component runs. Default: False.

Field	Description
Conversational	Indicate that there is a sequence of operations in the port type that must be called in order to achieve some higher level goal. Supported only when WS-Reliable Messaging is enabled for a SOAP binding. Default: False.
End Operation	The operation that signifies the end of the conversation. Enabled when Conversational is checked.

Bindings Tab

See [SOAP Binding Reference](#) and [JMS Binding Reference](#).

Policies Tab

[Policy Management](#)

Appearance Tab

Field	Description
Fonts and Colors	Specifies the font and color.
Lines and Arrows	Specifies the style of lines and arrows.
Gradient	Specifies the color gradient.

Bindings

A *Binding* specifies how communication happens between a Reference and a Service. A Service Binding describes the mechanism a client uses to access a Service. A Reference Binding describes the access mechanism a Reference uses to invoke a Service. References can have at most one Binding.

TIBCO ActiveMatrix supports the following Binding Types (BT):

- Virtualization
- REST
- SOAP
- JMS

Virtualization Bindings connect Services and References to the Messaging Bus. Virtualization Bindings are automatically created for every Composite Service and every wired component Service and Reference. At design-time, Virtualization Bindings of Component Services and References are implicit; their properties cannot be viewed.

There are two types of Virtualization Bindings: internal and external. An *internal binding* is associated with a Component Service or Reference. An *external binding* is associated with a Service or Reference promoted to the root composite. Administrators can create or modify wires connected to external bindings and can monitor, start, and stop external bindings.

The following bindings are explicitly created by architects and developers only on promoted services and references:


- SOAP
- JMS
- REST



TIBCO Business Studio and TIBCO ActiveMatrix Administrator provide the option to choose between TIBCO's SOAP/JMS and W3C SOAP/JMS for SOAP Binding Type while adding a Binding to a Service.

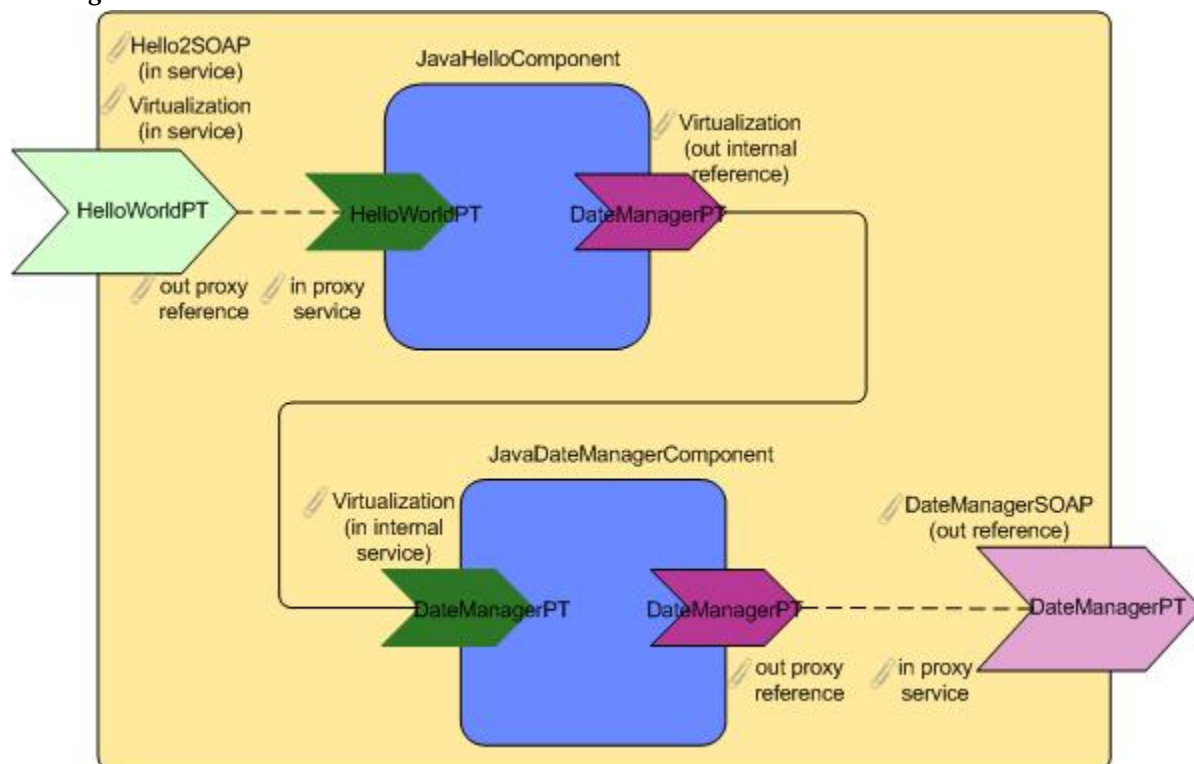


SOAP Bindings support both HTTP and JMS transport type.

The following figure, Bindings are indicated by a  icon. The promoted Service HelloWorldPT has a SOAP and external Virtualization Binding. The Components have internal Virtualization bindings. The promoted reference DateManagerPT has a SOAP binding. In addition, any time a Service or Reference has a binding of type other than Virtualization, a pair of proxy (Virtualization) bindings are created to connect the Service or Reference to the Component to which the Service or Reference Service is wired.

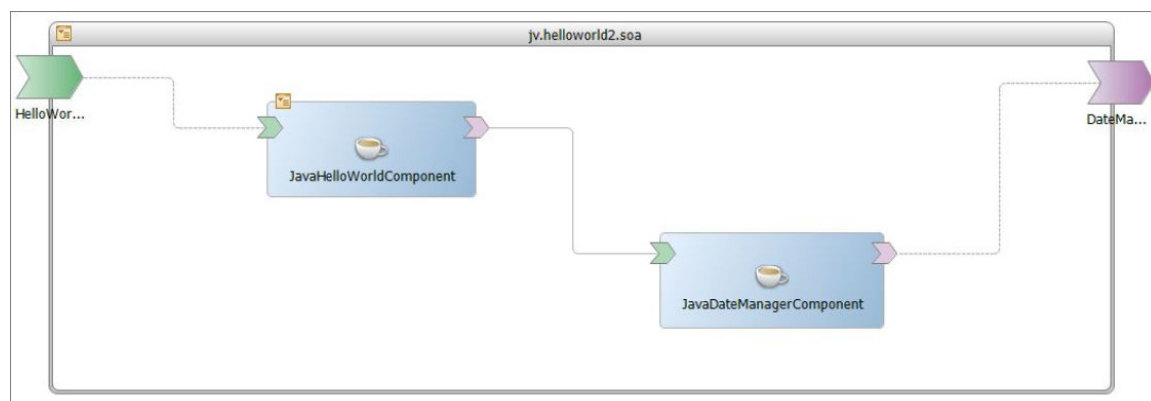
This figure is a representation of the TIBCO_HOME/samples/java/helloworld2.zip sample in the installation. You can run the sample to get a better understanding of the Bindings.

Bindings



Here is an example of how this scenario looks in TIBCO Business Studio:

Bindings in Business Studio



Viewing Service and Reference Bindings

Procedure

1. Click the service or reference.
2. In the Properties view, click the **Binding** tab.

Adding a Binding to a Service or Reference

You can add a binding to a service or reference from the canvas or from the Properties view.

Procedure

1. Choose a starting point and follow the appropriate procedure.

Starting Point	Procedure
Canvas	1. Right-click the service or reference and select Add > Binding...
Properties view	<ol style="list-style-type: none"> 1. Click the service or reference. 2. In the Properties view, click the Bindings tab. 3. Click Add Binding...

The Add Binding dialog displays.

2. Click a binding type.
3. Uncheck the **Generate Default Required Resource Templates** checkbox if you do not want resource templates required by the binding to be created.
4. Click **Finish**.
A binding is added to the list of bindings. If the Generate Default Required Resource Templates checkbox is checked, a resource named *RequiredResourceType_WSDLPortName.requiredResourceType* is created in the Resource Templates folder, and the binding property that requires the resource is configured with the generated resource.

SOAP Bindings

SOAP bindings serve as a gateway for inbound and outbound SOAP messages. SOAP bindings expose endpoints that accept requests from SOAP consumers and allow composites to invoke external SOAP providers.

When the request message's Accept-Encoding header is *gzip*, *deflate*, the SOAP Binding can also respond with a compressed response message.

SOAP bindings support the following features:

- [SOAP 1.1](#) and [SOAP 1.2](#) specifications.
- Encoding: Document-literal and RPC-literal
- Message exchange patterns: one-way, request-response, and fault
- Changing Endpoint URI for SOAP-HTTP Reference from Administrator UI and command-line interface.
- HTTP and JMS transport
- SOAP headers
- WS-Addressing
- WS-Reliable Messaging



If you change the order of operations in the WSDL interface of a service or reference you must recreate all SOAP bindings associated with the service or reference.

SOAP Binding Reference

You can specify the endpoint, SOAP defaults, service transport, and reference transport for the binding node. You can specify the SOAP general configuration for the operation node, and the part list for the input or output message node.

Binding Node

Binding

Field	Required?	Editable?	Description
Name	Y	Y	The name of the binding. Default: SOAPService_Binding n , where n is an integer.
Description	N	Y	A description of the binding. Default: None.
Target Namespace	Y	Y	The target namespace for a concrete WSDL file for the service. Default: <i>AbstractWSDLNamespace/BindingName</i> , where <i>AbstractWSDLNamespace</i> is the namespace of the abstract WSDL file of the service and <i>BindingName</i> is the value of the Name field.



SOAP Default Configuration

Field	Required?	Editable?	Description
SOAP Version	N	Y	The version of the SOAP specification: 1.1 or 1.2. Default: 1.1.
Style	Y	Y	The SOAP binding style: Document or RPC. Default: Document.
Encoding	N	N	The encoding type for the body of the SOAP input and output messages. Set to Literal.

Service Transport Configuration


Field	Required?	Editable?	Description
Transport Type	Y	Y	The type of transport supported by the binding. HTTP or JMS.
HTTP			
Endpoint URL	Y	Y	An absolute URL when its value contains an underscore (" _"), especially in the hostname part of the URL.

Field	Required?	Editable?	Description
Enable WS-Reliable Messaging	N	Y	Indicate whether to enable WS-Reliable Messaging behavior. When checked, the Quality of Service drop-down list displays. Default: Unchecked.
Quality of Service	N	Y	The level of guarantee for message delivery. Default: Exactly Once.
Enable WS-Addressing	N	Y	Indicate whether to enable WS-Addressing headers. When checked, the Anonymous checkbox displays. Default: Unchecked.
Anonymous	N	Y	Indicate whether synchronous responses should be returned to the client when WS-Addressing is enabled. Default:
Connector Name	Y	Y	The name of the HTTP connector resource instance that provides incoming transport services. Default: httpConnector.
Session Inactivity Timeout (s)	Y	Y	Specifies the time, in seconds, between the client requests before the servlet container invalidates the session. Default: 60.
JMS			
Binding Specification	Y	Y	Binding specification supported: TIBCO or W3C SOAP-JMS. Default: TIBCO. The thread pool parameters can be configured at a SOAP/JMS service binding level in addition to the node level. The parameters set on a SOAP/JMS service binding take precedence over the values defined at the node level as below: <code>java.property.com.tibco.amf.bindingtype.soap.endpointPoolSteadySize=10</code> <code>java.property.com.tibco.amf.bindingtype.soap.endpointPoolMaxSize=15</code> If neither sets of values are specified, then a system default of 10 and 15 are used for <code>endpointPoolSteadySize</code> and <code>endpointPoolMaxSize</code> respectively.
JMS - Request Configuration			
JMS Connection Factory	Y	Y	A JMS Connection Factory.

Field	Required?	Editable?	Description
JMS Destination	Y	Y	<p>A JMS destination configuration.</p>  Only queues are supported for SOAP/JMS. Topics are not supported.
Acknowledge Mode	Y	Y	The acknowledgment mode for incoming messages. Set to Auto, meaning that the message is automatically acknowledged when it is received.
Message Type	Y	Y	The type of the message content: Text or Bytes. Default: Text.
JMS - Reply Configuration			
JMS Connection Factory	Y	Y	A JMS Connection Factory .
Delivery Mode	N	Y	<p>The delivery mode of messages:</p> <ul style="list-style-type: none"> Persistent Messages are stored and forwarded. Non-Persistent Messages are not stored and may be lost due to failure. <p>Default: Persistent.</p>
Correlation Scheme	Y	Y	<p>Scheme which identifies the correlation scheme used when sending reply messages.</p> <ul style="list-style-type: none"> MessageID to CorrelationID (default) — Message ID of the request message is copied to the Correlation ID of the response message. CorrelationID to CorrelationID — Correlation ID of the request message is copied to the Correlation ID of the response message. Infer from Request — If CorrelationID is present in incoming Request Message, CorrelationID of incoming Request Message is copied to CorrelationID of outgoing Response Message. If CorrelationID is absent in incoming Request Message, MessageID of incoming Request Message (which is always present) is copied to CorrelationID of outgoing Response Message.  The correlation scheme is applicable only for SOAP/JMS service bindings. The correlation scheme is applicable on the reference side as well.


Field	Required?	Editable?	Description
Message Priority	N	Y	The priority of the message. Priority is a value from 0-9. Higher numbers signify a higher priority (that is, 9 is a higher priority than 8). Default: 4.
Message Expiration (ms)	N	Y	The length of time a message can remain active. 0 means that the message does not expire. Default: 0.

Reference Transport Configuration

Field	Required?	Editable?	Description
Transport Type	Y	N	The type of transport supported by the binding: HTTP or JMS.
HTTP			
HTTP Client	Y	Y	An HTTP Client .
Endpoint URI	Y	N	<p>The Endpoint URI field on reference is a combination of the hostname, port number and the Filespec or Endpoint URL. Though this field is not editable, the Filespec or Endpoint URL part can be changed by providing a complete URL or a URI.</p> <p>You can also pick or enter a substitution variable for the value, for example, %%MyPartnerEndPoint%%</p> <p>You can define the value for the substitution variable with a complete URL, for example, <code>http://<host>:<port>/weatherReportPT</code>.</p> <p>SSL-enabled SOAP/HTTP Endpoints</p> <p>You can override the protocol value in the Filespec or Endpoint URL field to <code>https://</code> to enable SSL. This requires pre-configuration of HTTP Client shared resource to use SSL Client Provider. Even if SSL is enabled in the HTTP Client shared resource, the protocol value in that field determines whether SSL is enabled or not.</p> <div>  <p>It is recommended that in TIBCO Business Studio you pick a substitution variable instead of entering it manually so that it is packaged correctly in the DAA and correctly handles the 'Synchronization' state in the Administrator.</p> </div>

Field	Required?	Editable?	Description
Endpoint URI Filespec	Y	Y	The endpoint URI filespec. This field is populated from the SOAP Address element of the WSDL port associated with the SOAP-HTTP reference binding. This value can be edited by typing the new value or by using the Substitution Variables picker to select a substitution variable that points to a valid endpoint URI value.
Override HTTP Client Socket Timeout	N	N	A boolean field check box representing whether the "Binding Socket Timeout" value is to be used (when true), or ignored (when false).
Binding Socket Timeout	N	N	An integer field representing a timeout value in milliseconds for a synchronous reply to arrive when an external service is invoked. It accepts substitution variables. This value is used only when the Override HTTP Client Socket Timeout is selected. This field is equivalent to the field Socket Timeout of the HTTP Client resource template.
Enable WS-Reliable Messaging	N	Y	Indicate whether to enable WS-Reliable Messaging behavior. When checked the Quality of Service drop-down list displays and the Enable WS-Addressing and Anonymous checkboxes are checked. Default: Unchecked.
Quality of Service	N	N	The level of guarantee for message delivery. Default: Exactly Once.
Enable WS-Addressing	N	Y	Indicate whether to enable WS-Addressing headers. When checked, the Anonymous checkbox displays and is checked. Default: Unchecked.
Anonymous	N	Y	Indicate whether synchronous responses should be returned to the client when WS-Addressing is enabled. When unchecked, the Connector Name field displays. Default: Checked.
Connector Name	N	Y	The name of the connector to which responses should be sent. Default: None.
JMS			
Binding Specification	Y	N	Binding specification supported: TIBCO or W3C SOAP-JMS. Default: TIBCO.
JMS - Request Configuration			

Field	Required?	Editable?	Description
Connection Factory	Y	Y	A JMS Connection Factory.
Destination	Y	Y	<p>A JMS destination configuration.</p> <p>Only queues are supported for SOAP/JMS. Topics are supported in a limited way; only SOAP/JMS reference bindings containing one-way operations (exclusively) with topics as destinations can be enabled by adding the following line to the node's TRA and restarting the node:</p> <pre>java.property.com.tibco.amf.bindingtype.soap.suppressTopicUnsupportedError=true</pre>
Delivery Mode	Y	Y	<p>The delivery mode of messages:</p> <ul style="list-style-type: none"> Persistent Messages are stored and forwarded. Non-Persistent Messages are not stored and may be lost due to failure. <p>Default: Persistent.</p>
Message Priority	Y	Y	<p>The priority of the message. Priority is a value from 0-9. Higher numbers signify a higher priority (that is, 9 is a higher priority than 8).</p> <p>Default: 4.</p>
Message Expiration (ms)	Y	Y	<p>The length of time a message can remain active. 0 means that the message does not expire.</p> <p>Default: 0.</p>
JMS - Reply Configuration			
Destination	Y	Y	<p>A JMS destination configuration.</p> <div>  <p>Only queues are supported for SOAP/JMS. Topics are not supported.</p> </div>
Acknowledge Mode	Y	N	<p>The acknowledgment mode for incoming messages. Set to Auto, meaning that the message is automatically acknowledged when it is received.</p>

Field	Required?	Editable?	Description
Correlation Scheme	Y	Y	<p>Scheme which identifies the correlation scheme used when sending reply messages.</p> <ul style="list-style-type: none"> MessageID to CorrelationID (default) — Message ID of the request message is copied to the Correlation ID of the response message. CorrelationID to CorrelationID — Correlation ID of the request message is copied to the Correlation ID of the response message. <div>  <p>The correlation scheme is applicable only for SOAP/JMS service bindings.</p> <p>The correlation scheme is applicable on the reference side as well.</p> </div>

Operation Node

SOAP General Configuration

Field	Required?	Editable?	Description
SOAP Action	Y	Y	The SOAP action that is expected from incoming SOAP requests. See the SOAP specification for more information about SOAP action.
Style	Y	Y	Document, RPC, or Inherit from parent endpoint.
Encoding	Y	Y	Literal.

Input or Output Message Node

Part List

Field	Description
Part Name	The name of the message part.
Part Type	The type of the message part: Body or Header.

Fault Messages

In service-oriented applications, SOAP clients expect a fault message to be returned when an error occurs during processing. A *fault message* is a SOAP message.

A fault message has the following subelements:

Subelement	Description
faultcode	A code that identifies the fault.

Subelement	Description
<code>faultstring</code>	An explanation of the fault.
<code>faultactor</code>	Information about what caused the fault to occur.
<code>detail</code>	Application-specific information about the fault.



A stack trace is included in the SOAP fault's `<detail>` element by default, but it can be suppressed by adding the following line in the runtime node's TRA file and restarting the node:

```
java.property.com.tibco.soapbt.spline.soapfaulttransformer.suppressstacktraceinsoapfault=true
```

The SOAP fault messages for a SOAP/HTTP reference binding timeout provides contextual information. When a timeout occurs on a partner call made from the application, the following additional information is provided in the SOAP Fault message, inside the 'faultstring' element:

- The Binding (Endpoint) Name and Promoted Reference Name
- Node name of the Reference Binding
- Target URL that resulted in the timeout
- Timeout Value specified and in effect
- Time at which Timeout occurred
- Service (Endpoint) Name and Promoted Service Name (where the call originated)
- Host, Port and Endpoint URI of the Service Binding
- Names of the Application, Node and Environment (of the Service Binding)
- Name of the Operation invoked
- Host Address of the originating client (that sent the request to the Service Binding)

Fault messages defined in the WSDL file are called *declared faults*. Fault messages that are not defined in the WSDL file are called *undeclared faults*. The process for generating a fault message is implementation dependent and typically depends on whether the fault is declared or not.

Declared Faults

A service can define multiple fault messages for any operation. When an error occurs, it will return a fault message to indicate the specific error.

Example WSDL File

The following WSDL fragment shows the `getWeather` operation with two declared faults: `orderFault` and `orderFault2`. The detail element for the `orderFault` message contains a `ZipCodeFault` element. The detail element for the `orderFault2` message contains a `CityFault` element.

```
<wsdl:types>
  <schema xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:tns="http://www.example.org/weatherschema"
    targetNamespace="http://www.example.org/weatherschema"
    elementFormDefault="unqualified"
    attributeFormDefault="unqualified">
    <complexType name="WeatherRequestType">
      <sequence>
        <element name="city" type="string"/>
        <element name="state" type="string"/>
        <element name="zip" type="string"/>
      </sequence>
    </complexType>
    <complexType name="WeatherResponseType">
      <sequence>
        <element name="high" type="float"/>
        <element name="low" type="float"/>
        <element name="forecast" type="string"/>
      </sequence>
    </complexType>

    <element name="WeatherRequest" type="tns:WeatherRequestType"/>
    <element name="WeatherResponse" type="tns:WeatherResponseType"/>
    <element name="ZipCodeFault" type="string"/>
    <element name="CityFault" type="string" />
  </schema>
</wsdl:types>
<wsdl:message name="invalidZipCodeFault">
  <wsdl:part name="error" element="ns0:ZipCodeFault"/>
</wsdl:message>
<wsdl:message name="invalidCityFault">
  <wsdl:part name="error" element="ns0:CityFault" />
</wsdl:message>
<wsdl:portType name="WeatherReportPT">
  <wsdl:operation name="GetWeather">
    <wsdl:input message="tns:GetWeatherRequest"/>
    <wsdl:output message="tns:GetWeatherResponse"/>
    <wsdl:fault name="orderFault" message="tns:invalidZipCodeFault"/>
    <wsdl:fault name="orderFault2" message="tns:invalidCityFault" />
  </wsdl:operation>
</wsdl:portType>
```

Example Declared Fault Message

The following is a sample of an `invalidZipCodeFault` message. The SOAP Fault element contains error information within a SOAP message.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
envelope/">
<SOAP-ENV:Body>
<SOAP-ENV:Fault>
  <faultcode xmlns="">SOAP-ENV:soap:Server</faultcode>
  <faultstring xmlns="">Could not find the zip code</faultstring\>
  <faultactor xmlns="">http://www.weather.com/forecast</faultactor>
  <detail xmlns="">
    <ns:ZipCodeFault xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:ns="http://www.example.org/weatherschema"
      xmlns:ns0="http://schemas.xmlsoap.org/soap/envelope/">94305</
ns:ZipCodeFault>
  </detail>
</SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Undeclared Faults

Undeclared faults are handled in an implementation-dependent fashion.

Example Undeclared Fault Message

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
envelope/">
<SOAP-ENV:Body>
  <SOAP-ENV:Fault>
    <faultcode>SOAP-ENV:Server</faultcode>
    <faultstring>java.lang.RuntimeException: Undeclared fault....</
faultstring>
    <faultactor>DefaultRole</faultactor>
    <detail>
      <tibco:myFaultDetail xmlns:tibco="http://tibcouri/">
        org.osoa.sca.ServiceRuntimeException:
java.lang.RuntimeException: Undeclared fault....
        at
com.tibco.amf.platform.runtime.componentframework.internal.proxies.
ProxyInvocationHandler.invoke(ProxyInvocationHandler.java:473)
        at $Proxy21.invoke(Unknown Source)
        at
com.tibco.amf.binding.soap.runtime.transport.http.SoapHttpInboundEndp
oint.
        processHttpPost(SoapHttpInboundEndpoint.java:250)
        at
com.tibco.amf.binding.soap.runtime.transport.http.SoapHttpServer.doPo
st(
        SoapHttpServer.java:103)
        ...
        Caused by: java.lang.RuntimeException: Undeclared fault....
        at
com.sample.faultservice.Component1.getWeather(Component1.java:50)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.
java:39)
        at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAcces
sorImpl.java:25)
        at java.lang.reflect.Method.invoke(Method.java:585)
        at
com.tibco.amf.platform.runtime.componentframework.internal.proxies.
ProxyInvocationHandler.invoke(ProxyInvocationHandler.java:426)
        ... 20 more
      </tibco:myFaultDetail>
    </detail>
  </SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

WS-Addressing

You can use WS-Addressing to specify message destinations and other information that is not part of the SOAP protocol.

The SOAP protocol does not provide a standard way to specify message destination, where to return a response, or how to report an error. Traditionally these details have usually been handled by the transport layer. For example, when a SOAP request is sent over HTTP, the URI of the HTTP request determines the message's destination. The message response is packaged in the HTTP response and received by the client over the HTTP connection. When a SOAP request message is sent asynchronously through JMS, a destination for responses might be specified in the JMS message headers, incorporated into the message body, or left up to the service implementation.

The [WS-Addressing specifications](#) provide a uniform method for incorporating delivery, reply-to, and fault handler addressing information into a SOAP envelope. WS-Addressing defines two constructs that convey such addressing information: endpoint references and message addressing properties.

Endpoint References

An *endpoint reference* conveys the information needed to identify a Web service endpoint, but are also used to provide addresses for individual messages sent to and from Web services. An endpoint reference contains an address (a URI), reference parameters, and metadata.

For details on endpoint references, refer to the [WS-Addressing Core Specification](#). For information on WS-Addressing, see *Composite Development*.

Schema

The schema of an endpoint reference is described in [Web Services Addressing 1.0 - Core: Endpoint Reference XML Infoset Representation](#).

URIs

The only required element of an endpoint reference is the address; the other elements are optional. Thus, the simplest endpoint reference is a URI:

```
<wsa:Address>
http://localhost:9090/axis2/services/billing_service
</wsa:Address>
```

The supported URI formats are listed in the following table:

Supported URI Formats

Binding Type	URI Format
Virtualization	urn:amx:environmentName/applicationName#service(serviceName) where <i>serviceName</i> is the name of the target service, <i>applicationName</i> is the name of the application that contains the target service, and <i>environmentName</i> is the name of the ActiveMatrix environment that contains the application.
SOAP/HTTP	<i>scheme</i> ://hostname:port/filespec/
SOAP/JMS	jms:queue:queueName, where <i>queueName</i> is the name of the JMS queue to which messages are sent.

Reference Parameters

A reference parameter is associated with an endpoint to facilitate a particular interaction. The binding of reference parameters to messages depends upon the protocol binding used to interact with the endpoint. [Web Services Addressing 1.0 - SOAP Binding](#) describes the default binding for the SOAP protocol.

Metadata

Endpoint reference metadata describes the behavior, policies, and capabilities of the endpoint. Metadata may be included in an endpoint reference to facilitate easier processing by a user of an endpoint reference, or because the metadata was dynamically generated.

Example

```
<wsa:EndpointReference
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wsaw="http://www.w3.org/2006/02/addressing/wsdl"
  xmlns:fabrikam="http://example.com/fabrikam"
  xmlns:wsdli="http://www.w3.org/2006/01/wsdl-instance"
  wsdli:wsdlLocation="http://example.com/fabrikam
    http://example.com/fabrikam/fabrikam.wsdl">
  <wsa:Address>http://example.com/fabrikam/acct</wsa:Address>
  <wsa:Metadata>
    <wsaw:InterfaceName>fabrikam:Inventory</wsaw:InterfaceName>
  </wsa:Metadata>
  <wsa:ReferenceParameters>
    <fabrikam:CustomerKey>123456789</fabrikam:CustomerKey>
    <fabrikam:ShoppingCart>ABCDEFG</fabrikam:ShoppingCart>
  </wsa:ReferenceParameters>
</wsa:EndpointReference>
```

Message Addressing Properties

Message addressing properties allow uniform addressing of messages independent of underlying transport. These properties convey end-to-end message characteristics including addressing for source and destination endpoints as well as message identity.

Most of the properties are optional; the only required property is the action property. The value of the [action] property should be an IRI identifying an input, output, or fault message within a WSDL interface or port type. An action may be explicitly or implicitly associated with the corresponding WSDL definition. See [Configuring the Action Property](#). If present, the request is delivered to the IRI specified in the To element. The action IRI indicates the action to be taken. In an HTTP request, these would be the same IRI. In a non-HTTP request, the To IRI may differ from the action IRI.

Message Addressing Properties

The message addressing properties augment a message with the properties listed in [Message Addressing Properties](#).

Message Addressing Elements

The syntax of the message addressing elements is described in [Message Addressing Elements](#).

Example

```
<soapenv:Envelope xmlns:soapenv='http://www.w3.org/2003/05/soap-
  envelope'>
  <soapenv:Header xmlns:wsa='http://www.w3.org/2005/08/addressing'>
    <wsa:To>http://localhost:9090/axis2/services/order_service</wsa:To>
    <wsa:Action>http://tibco.com/addr/order</wsa:Action>
    <wsa:ReplyTo>
      <wsa:Address>http://localhost:9090/axis2/services/
        billing_service</wsa:Address>
    </wsa:ReplyTo>
    <wsa:FaultTo>
      <wsa:Address>http://localhost:9090/axis2/services/
        reorder_service</wsa:Address>
    </wsa:FaultTo>
    <wsa:MessageID>a4dfb94a-593b-1dc1-36d2-000000000000</wsa:MessageID>
  </soapenv:Header>
  <soapenv:Body>
  </soapenv:Body>
</soapenv:Envelope>
```

Validation and Fault Handling

Most WS-Addressing elements are optional. If these elements are omitted, the SOAP binding does not return a fault message. To enable validation, enable the endpoint for WS-Addressing.

The faults defined in this section are generated if the condition stated in the preamble in each subsection is met. They are sent to the [fault endpoint], if present and valid. Otherwise they are sent to the [reply endpoint] if present. If neither is present faults may be sent to the [source endpoint].

The [action] property designates WS-Addressing fault messages. This URI is also used as default Action value for WSDL fault messages: <http://schemas.xmlsoap.org/ws/2004/08/addressing/fault>

The definitions of faults use the following properties:

Property	Description
[Code]	The fault code.
[Subcode] or [Subsubcode]	The fault subcode.
[Details]	The detail element. If absent, no detail element is defined for the fault.

SOAP Binding Flags

The handling of WS-Addressing headers depends on the state of the Enable WS-Addressing SOAP binding flag. When checked, the WS-Addressing headers are validated. If unchecked, the request URI determines the service name and the soapAction transport header determines the operation. The following sections describe the behavior of the SOAP binding when incoming requests are missing WS-Addressing header elements.

/wsa:Action

- Missing /wsa:Action The SOAP container returns a fault message with the details:

```
<ProblemAction xmlns="http://www.w3.org/2005/08/addressing">
  <Action>NULL</Action>
</ProblemAction>
```

If this header is set but is invalid, the SOAP binding returns a fault message with the details:

```
<ProblemAction xmlns="http://www.w3.org/2005/08/addressing">
  <Action>invalidAction</Action>
</ProblemAction>
```

/wsa:To

- Missing /wsa:To

The value of the [destination] property is set to <http://www.w3c.org/2005/08/addressing/anonymous>.

- Missing Both /wsa:Action and /wsa:To

The SOAP binding returns a fault message with the details:

```
<ProblemAction xmlns="http://www.w3.org/2005/08/addressing">
  <Action>NULL</Action>
</ProblemAction>
```

/wsa:ReplyTo

- Missing /wsa:ReplyTo

The [address] property of the [reply endpoint] is set to `http://www.w3c.org/2005/08/addressing/anonymous`. If this element is missing or invalid, the SOAP container synchronously returns a response message to the client with WS-Addressing header populated.

- Missing `/wsa:ReplyTo/Addressing`

If `/wsa:ReplyTo` element is present, but is missing the required address subelement, the SOAP container returns a fault message with subcode=`InvalidAddressingHeader` and subsubcode=`MissingAddressInEPR`. The [action] property is set to `http://www.w3.org/2005/08/addressing/soap/fault`.

/wsa:FaultTo

- Missing `/wsa:FaultTo`

If the reply is a fault message the [reply endpoint] property is used.

- Missing Both `/wsa:FaultTo` and `/wsa:ReplyTo`

The response is sent back to the client directly. The [action] property is set to `http://www.w3.org/2005/08/addressing/soap/fault`.

- Missing `/wsa:FaultTo/Addressing`

If the `/wsa:FaultTo` element is present, but is missing the required Address subelement, the SOAP container returns a fault message with subcode `InvalidAddressingHeader` and subsubcode=`MissingAddressInEPR`. The [action] property is set to `http://www.w3.org/2005/08/addressing/soap/fault`.

- Missing `/wsa:MessageID`

The SOAP container returns a fault message with subcode=`MessageAddressingHeaderRequired`.

Configuring the Action Property

WS-Addressing defines two mechanisms to associate a value of the [action] property with input, output, and fault elements within a WSDL description: explicit association and default association.

Explicit Association

In an explicit association, the [action] property value is set from the value of the `Action` elements specified for the input, output, and fault messages or the value of the `soapAction` attribute set in the transport header.

```
<?xml version="1.0" encoding="utf-8"?>
<definitions targetNamespace="someuri">
  <portType name="Hello_PortType">
    <operation name="sayHello">
      <input message="SayHelloRequest" wsam:Action="http://tibco.com/HelloService/Request"/>
      <output message="SayHelloResponse" wsam:Action="http://tibco.com/HelloService/Response"/>
    </operation>
  </portType>
```

```
Input message [action] = "http://tibco.com/HelloService/Request"
Output message [action] = "http://tibco.com/HelloService/Response"
```

Default Association

If neither the `Action` elements or `soapAction` attribute is specified, the [action] property value is constructed as follows:

- **Input and output messages**

targetnamespace/porttypename/messagename

- **Fault message**

targetnamespace/porttypename/operationname/Fault/messagename

```
<?xml version="1.0" encoding="utf-8"?>
<definitions targetNamespace="http://tibco.com/defaulting ">
<portType name="Hello_PortType">
  <operation name="sayHello">
    <input message="SayHelloRequest"/>
    <output message="SayHelloResponse" />
    <fault message="InvalidMessage" name="InvalidRequest"/>
  </operation>
</portType>

Input message [action] =
"http://tibco.com/defaulting/ Hello_PortType/SayHelloRequest"
Output message [action] =
"http://tibco.com/defaulting/ Hello_PortType/SayHelloResponse"
Fault message [action] =
"http://tibco.com/defaulting/Hello_PortType/ sayHello/Fault/InvalidRequest"
```

Enabling WS-Addressing

Procedure

1. Click a reference or service.
2. In the Properties view, click the **Bindings** tab.
3. Choose a starting point and follow the appropriate procedure.

Starting Point	Procedure
Binding Node	<ol style="list-style-type: none"> 1. Click the binding node of a SOAP binding. 2. Check the Enable WS-Addressing checkbox in the Transport Configuration area.
Policy Sets	<ol style="list-style-type: none"> 1. Click the Policy Sets node. 2. Add a WS-Addressing policy to the binding's embedded policy set.

The two procedures have equivalent results. If you check the checkbox, the policy set is added to the Policy Sets > Binding Policy Sets node and vice versa.

Generating a Concrete WSDL File

Procedure

1. Click a promoted service.
2. In the Properties view, click the **Bindings** tab.
3. Click the endpoint node of a SOAP binding.
4. In the Endpoint area on the right, click the **Generate WSDL** link.
The WSDL Generation dialog displays.
5. In the Enter or select the parent folder field, accept the default folder or select a new one.
6. In the File name field, accept the default name (*abstractWSDLFilename_gen.wsdl*) or type a new one.
7. Click **Next** >.
The Concrete WSDL Settings screen displays. The configuration fields depend on the value selected in the endpoint's Transport Type drop-down list.

8. Accept the default values of the transport properties or type new ones and click **Finish**.
A concrete WSDL file is generated in the parent folder and the file is opened in the WSDL file editor.

WSDL Generation Reference

Field	Description
HTTP Transport	
Host	Host portion of the location attribute of the <code>soap:address</code> element. Default: localhost.
Port	Port portion of the location attribute of the <code>soap:address</code> element. Default: 80.
Scheme	The scheme of the URI: http or https. Default: http.
JMS Transport	
Connection Factory	Connection factory. Default: JMSConnectionFactory.
Destination Type	Type of destination: Queue or Topic. Default: Queue.
Destination Name	Name of the destination. Default: sample.queue.
Common	
Namespace URI	Namespace of the generated WSDL file. Default: The value specified in the Target Namespace field of the SOAP binding.
Local Name	Name of the <code>service</code> element in the generated WSDL file. Default: The name of the service's port type.

Generating a SOAP Request

To test services with SOAP bindings, you can use the Web Services Explorer to generate SOAP requests. If request generation succeeds, a response is returned in the Status area.

Prerequisites

Web Services Explorer requires the use of an internal or external web browser. On Linux platforms, the internal browser is not enabled and sometimes the Web Service Explorer is not able to find the list of installed external browsers. To add an installed browser:

1. Select **Window > Preferences > General > Web Browser**
2. Check the **Use external web browser** radio button.

3. Click **New**.
4. In the Name field, type a browser name.
5. In the Location field, type the path to a browser executable or click **Browse...** and navigate to a folder containing the browser executable.
6. After selecting the browser executable, click **OK**.
7. Check the checkbox next to the browser that you added in the preceding step.
8. Click **Apply** and click **OK**.

For background information on the Web Services Explorer, see **Help > Help Contents > Supplemental Eclipse Help > Web Application Development User Guide > Developing Web service applications > Testing and validating Web services**.

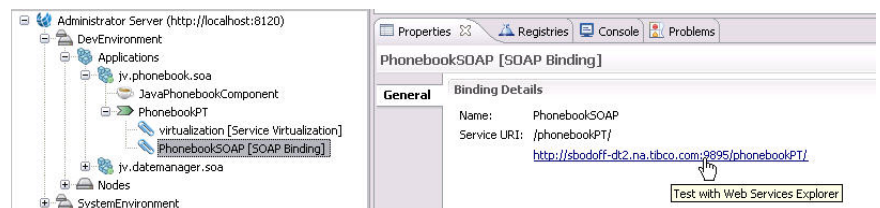
The following limitations apply when generating SOAP requests:

- The SOAP binding transport type must be HTTP; JMS transport is not supported.
- All message part types defined in the service's WSDL interface must be specified with an `element` attribute; a part type specified with a `type` attribute is not supported.

Procedure

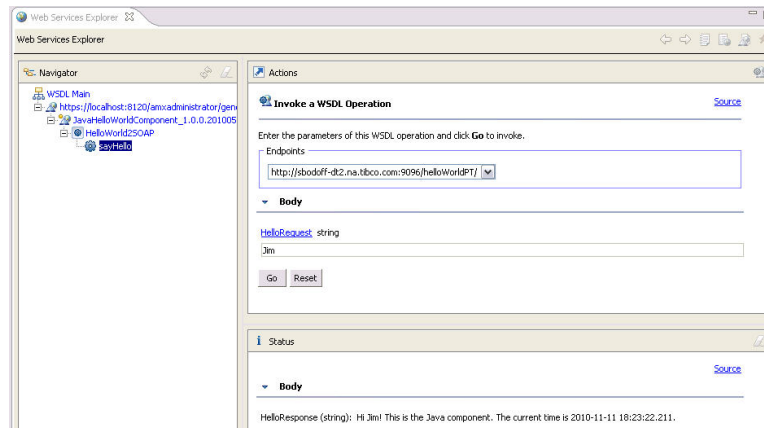
1. Choose a view and follow the appropriate procedure.

View	Procedure
Project Explorer	<ol style="list-style-type: none"> 1. Right-click a concrete WSDL file and select Web Services > Test with Web Services Explorer.
Composite Applications (local node)	<ol style="list-style-type: none"> 1. Expand the configuration node. 2. Right-click a SOAP Endpoint and select Test with Web Services Explorer.
Administrator Explorer (remote node)	<ol style="list-style-type: none"> 1. Expand the nodes of the Administrator server that manages the application. 2. Do one of the following: <ul style="list-style-type: none"> • Right-click a binding and select Test with Web Services Explorer. • <ol style="list-style-type: none"> 1. Click a binding. 2. In the Properties view, click the link below the Service URI field.



The WSDL file is opened in the Web Services Explorer and the binding is selected.

2. In the Navigator on the left, expand the binding node.
3. Click an operation in the Navigator pane.
4. In the Actions pane on the right, type a value for any required request parameters and click **Go**. The response is returned in the Status area.

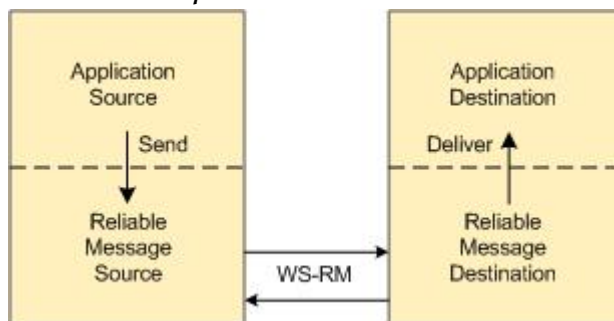


WS-Reliable Messaging

The OASIS Web Services Reliable Messaging 1.1 Specification describes a protocol that allows reliable message transfer in the presence of software component, system, or network failures. The specification describes the protocol in a transport-independent manner so it can be implemented using different network technologies. To support interoperable Web services, a SOAP binding is defined within the specification.

The participants in reliable messaging are application source (AS), application destination (AD), reliable message source (RMS), and reliable message destination (RMD), as shown in the following illustration.

WS-RM Participants



An AS wants to reliably send messages to an AD over an unreliable infrastructure. To accomplish this it uses a reliable message source (RMS) and a reliable message destination (RMD). The AS sends a message to the RMS. The RMS uses the WS-Reliable Messaging (WS-RM) protocol to transmit the message to the RMD. The RMD delivers the message to the AD. If the RMS cannot transmit the message to the RMD for some reason, raises an exception or otherwise indicates to the AS that the message was not transmitted. The AS and RMS can be implemented within the same process space or they be separate components. Similarly, the AD and RMD can exist within the same process space or be separate components.

Delivery Guarantees

WS-Reliable Messaging defines the following delivery guarantees:

- **At Least Once** Each message is delivered to the AD at least once. If a message cannot be delivered, an error must be raised by the RMS, RMD or both. Messages may be delivered to the consumer more than once (that is, the consumer may get duplicate messages).
- **At Most Once** Each message is delivered to the AD at most once. Messages may not be delivered to the AD, but the AD never gets duplicate messages.

- **Exactly Once** Each message is delivered to the AD exactly once. If a message cannot be delivered, an error must be raised by the RMS, RMD, or both. The AD never gets duplicate messages.
- **In Order** Messages are delivered from the RMD to the AD in the order that they are sent from the AS to the RMS. This guarantee can be combined with any of the other guarantees.

TIBCO ActiveMatrix supports Exactly Once delivery guarantee.

Composition with WS-Addressing

When the WS-RM protocol is composed with the WS-Addressing specification, the following rules prescribe the constraints on the value of the wsa:Action header:

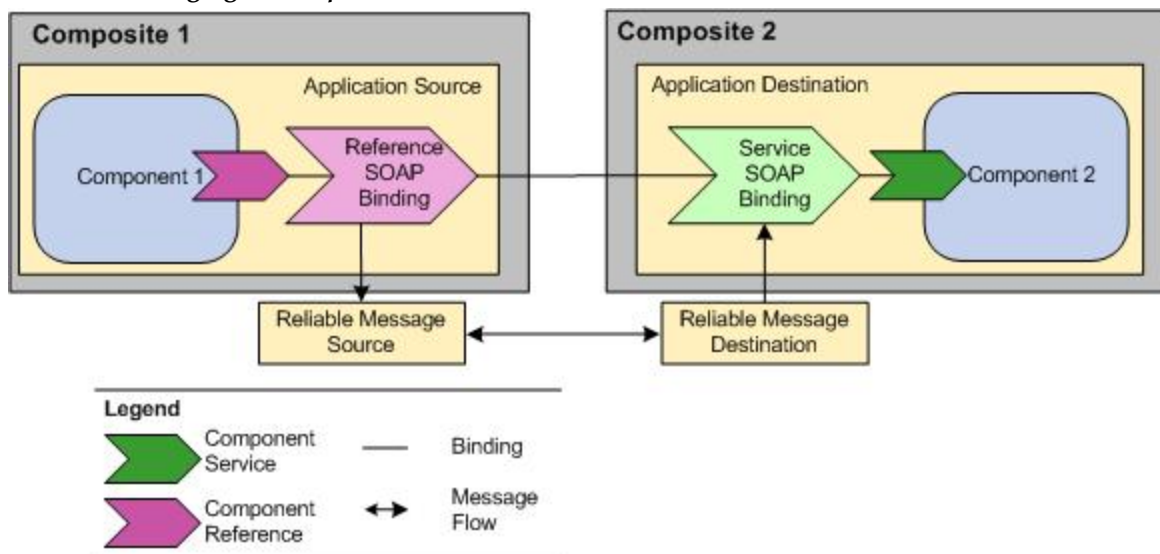
- When an endpoint generates a message that carries an RM protocol element in the body of a SOAP envelope, that endpoint must include in that envelope a wsa:Action SOAP header block whose value is an IRI that is a concatenation of the WS-RM namespace URI, followed by a "/", followed by the value of the local name of the child element of the SOAP body. For example, for a Sequence creation request message, the value of the wsa:Action IRI would be <http://docs.oasis-open.org/ws-rx/wsrn/200702/CreateSequence>.
- When an endpoint generates an acknowledgement message that has no element content in the SOAP body, then the value of the wsa:Action IRI must be <http://docs.oasis-open.org/ws-rx/wsrn/200702/SequenceAcknowledgement>.
- When an endpoint generates an acknowledgement request that has no element content in the SOAP body, then the value of the wsa:Action IRI must be <http://docs.oasis-open.org/ws-rx/wsrn/200702/AckRequested>.
- When an endpoint generates an RM fault, the value of the wsa:Action IRI must be <http://docs.oasis-open.org/ws-rx/wsrn/200702/fault>.

Reliable Messaging Elements

[noPageCitation](#) shows the four participants in a reliable messaging scenario.

[Reliable Messaging Elements](#) illustrates how the participants are mapped to composite elements. This section describes how to enable reliable messaging in the participating elements.

Reliable Messaging Participants



As shown in the figure, the Application Source role is performed by Component 1 and a SOAP reference.

Reliable messaging commences when Component 1 initiates a reliable conversation.

In order for the SOAP reference to participate in reliable messaging you must enable WS-Reliable Messaging for the reference. When reliable messaging is enabled, the SOAP reference communicates with a Reliable Message Source implemented by the platform.

The Application Destination role is performed by a SOAP service, which like the reference, must be enabled for WS-Reliable Messaging and Component 2. The SOAP service communicates with a Reliable Message Destination implemented by the platform.

Because WS-Reliable Messaging requires WS-Addressing, you must also enable WS-Addressing on both the SOAP reference and service.

Enabling WS-Reliable Messaging

Prerequisites

WS-Reliable Messaging requires WS-Addressing to be enabled.

Procedure

1. Click a reference or service.
2. In the Properties view, click the **Bindings** tab.
3. Choose a starting point and follow the appropriate procedure.

Starting Point	Procedure
Binding Node	<ol style="list-style-type: none"> 1. Click the binding node of a SOAP binding. 2. Check the Enable WS-Reliable Messaging checkbox in the Transport Configuration area of the binding node of a SOAP binding.
Policy Sets	<ol style="list-style-type: none"> 1. Click the Policy Sets node. 2. Add a WS-Reliable Messaging policy to the binding's embedded policy set.

The two procedures have equivalent results. If you check the checkbox, the policy set is added to the Policy Sets > Binding Policy Sets node and vice versa.

REST Bindings

REST bindings are completely integrated in ActiveMatrix now, and can be accessed and deployed like SOAP and JMS bindings. For more information on the REST Binding, refer to the *REST Binding Development Guide*.

JMS Bindings

JMS bindings integrate JMS applications with TIBCO ActiveMatrix. The JMS bindings convert JMS messages to TIBCO ActiveMatrix messages and vice versa.

[Java Message Service \(JMS\)](#) is a Java specification for messaging between applications. JMS is based on the creation and delivery of messages. The creator of the message is known as the publisher and the receiver of the message is known as the subscriber. A JMS server acts as an intermediary for the message and manages its delivery to the correct destination.

Configuration Overview

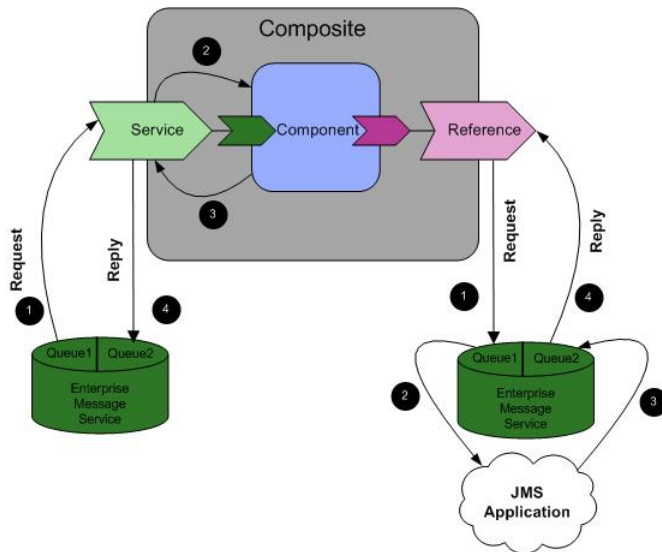
JMS bindings enable you to establish request and response message communication with a JMS server. In other words, adding a JMS binding enables a particular application to receive JMS messages or to send messages to the JMS server (JMS destination).

For an application to receive messages, for example, it must subscribe to a JMS server on a destination, which is defined by the JMS Connection Factory Configuration, JMS Destination Configuration, and JNDI Connection resource instances.

For the application to send messages, configuration details must be provided for the runtime library through the JMS Connection Factory, JMS Destination, and JNDI Connection resource instances.

The following figure illustrates an example of the request and response message communication sequence of a service and of a reference within a TIBCO ActiveMatrix component.

Service and Reference Request and Reply Communication



The communication sequence for the service, which corresponds to the numbers shown in the figure, is:

1. The service gets a message from the destination specified by the request destination.
2. The message is processed and sent to the component implementation.
3. If a response is received from the component implementation, and an incoming message was configured for a JMSReplyTo destination--either a temporary one or one specified as a service outbound destination--then the output goes to that destination.
4. The destination receives the message.

The communication sequence for the reference, which corresponds to the numbers shown in the figure, is:

1. A message is sent by the reference to a destination specified by the outbound destination.
2. Once the message goes to the destination, there is another application listening to that message.
3. The application gets the message and puts a response to JMSReplyTo specified on the incoming message--either a temporary one or one specified as a request destination.
4. The reference listens for responses on that destination and then receives one (4).

Use Cases

TIBCO ActiveMatrix supports the following JMS use cases and corresponding MEPs:

- **Service binding** - You can create a service referencing port types of a component hosted inside TIBCO ActiveMatrix. The component hosted inside TIBCO ActiveMatrix dictates the WSDL file and provides services.
 - TIBCO ActiveMatrix subscriber communicating with a JMS publisher - In-Only

- TIBCO ActiveMatrix server communicating with a JMS requestor - In-Out
- **Reference binding-** You can create a reference for endpoints in an existing JMS application. The JMS application dictates the WSDL file and provides services.
 - TIBCO ActiveMatrix publisher communicating with a JMS subscriber - In-Only
 - TIBCO ActiveMatrix client communicating with a JMS responder - In-Out


JMS Binding Reference

JMS Bindings include properties. You can configure most properties, and several properties accept substitution variables.

Binding Node




Property	Editable?	Accepts Svar?	Description
Name	Y	N	Name of JMS Binding.
Description	Y	N	Description of JMS Binding.
Connection Factory	Y	N	<p>The name of a JMS Connection Factory.</p> <p>Required for MEP:</p> <ul style="list-style-type: none"> • In-Out (Service, Reference) • In-Only (Service, Reference)

Configuration for JMS Binding Request Communication



Property	Editable?	Accepts Svar?	Description
Destination Type	Y	N	<p>The Type of JMS destination, Queue, Topic, or JNDI. For Direct Destinations, use Queue or Topic. For JNDI Resource template, use JMS Destination Resource template.</p> <p>Required for MEP:</p> <ul style="list-style-type: none"> • In-Only (Service, Reference) • In-Out (Service, Reference)
Destination	Y	N	<div>  <p>This property is only applicable for JNDI Destination Type.</p> </div> <p>The name of a JMS Destination in case of JMS Destination resource template.</p> <p>Required for MEP:</p> <ul style="list-style-type: none"> • In-Only (Service, Reference) • In-Out (Service, Reference)




Property	Editable?	Accepts Svar?	Description
Queue Name	Y	Y	Name of the Queue if Destination Type is selected as Queue.
Topic Name	Y	Y	Name of the Topic if Destination Type is selected as Topic.

Configuration for Reply JMS message, applicable for In-Out MEP

Property	Editable?	Accepts Svar?	Description
Destination Type	Y	N	<p>The Type of JMS destination, Queue, Topic or JNDI. For direct destinations use Queue or Topic. For JNDI resource template, use JMS Destination Resource template.</p> <p>By default destination type is 'Same as Request Message'.</p> <div>  <p>The 'Same as Request Message' option indicates that the Reply Message Destination Type is same as the Request Message Destination Type. In CLI script, there is no such option. You must select Queue, Topic, or JNDI.</p> </div> <p>Required MEP:</p> <ul style="list-style-type: none"> In-Out (Service, Reference)
Destination	Y	N	<div>  <p>This property is only applicable for JNDI Destination Type.</p> </div> <p>The name of a JMS Destination in case of JMS Destination resource template. If not specified, temporary destination name derived from value of JMSReplyTo JMS header will be used.</p>
Queue Name	Y	Y	Name of the Queue if Destination Type is selected as Queue.
Topic Name	Y	Y	Name of the Topic if Destination Type is selected as Topic.
<div>  <p>In case of In-Out MEP even when Reply Message is configured, priority will be given to JMSReplyTo JMS Message header and reply will be sent on the destination represented by the JMSReplyTo header value. Clients must not set this header field when fixed reply destination is used.</p> </div>			


Advanced Settings for JMS Binding


Property	Editable?	Accepts Svar?	Description
Reply Message NOTE: If Request or Reply message destination type is set to Queue or Topic and JMS Provider does not support dynamic queue or topic creation or the user of provider does not have create permissions, create a queue or topic before deploying the application.			
Connection Factory	Y	N	<p>Name of the JMS Connection Factory resource template.</p> <p>By default Connection Factory is 'Same as Request Message'.</p> <div>  <p>The 'Same as Request Message' option indicates that the Reply Message Connection Factory is same as the Request Message Connection Factory. In CLI script, there is no such option.</p> </div> <p>Required MEP:</p> <ul style="list-style-type: none"> In-Out (Service, Reference)
Correlation Scheme	Y	Y	<p>Scheme which identifies the correlation scheme used when sending reply messages.</p> <p>Required if the reply destination is set. The correlation schemes are:</p> <ul style="list-style-type: none"> RequestCorrelIDtoCorrelID - Correlation ID of the request message is copied to the Correlation ID of the response message. RequestMsgIDtoCorrelID - Message ID of the request message is copied to the Correlation ID of the response message. <p>For receiving proper reply messages by the JMS Binding on Promoted Reference in case of In-Out MEP, to pick the message from Request Destination, client must set the JMSCorrelationID header field on the JMS Message according to the Correlation Scheme.</p> <p>Default: RequestCorrelIDtoCorrelID</p> <div>  <p>RequestMsgIDtoCorrelID correlation scheme is not supported for Topic set as static reply destination.</p> </div>
Operation Selection			


Property	Editable?	Accepts Svar?	Description
Type	Y	N	 <p>Applicable only in case of multiple operations.</p> <p>Operation selection scheme in case of multiple operations. SCA and Custom are supported. In case of Custom scheme other properties (JMS Property Name and Error Action) are not editable but Message Selector configuration on each operation is mandatory. See "Operation Node" for more details.</p>
JMS Property Name	Y	Y	 <p>Applicable only in case of multiple operations.</p> <p>Name of the JMS property to be used for operation selection in case of multiple operation and "SCA" operation selection type. Default property name is "scaOperationName".</p>
Error Action	Y	N	 <p>Applicable only in case of multiple operations.</p> <p>Action to trigger in case when operation selection from multiple operation fails.</p> <ul style="list-style-type: none"> • Discard Message - This is a default Error Action. When selected, runtime will discard the message when operation selection fails. • Send Message To Operation - By selecting this action, user can inform runtime to send the message to a particular configured operation when operation selection fails. • Send Message To Error Queue - By selecting this action, user can inform runtime to send the message to a configured error queue when operation selection fails. • Retain Message in Service Destination - By selecting this action, user can inform runtime to retain the message in the service request destination configured in Request Message section.
Operation Name	Y	Y	<p>Displayed when "Send Message to Operation" error action is selected. Operation name to send the message in case of operation selection fails and "Send Message to Operation" is configured.</p>
Error Queue Name	Y	Y	<p>Displayed when "Send Message to Error Queue" error action is selected. Error queue to send the JMS message in case of operation selection fails and "Send Message to Error Queue" error action is configured.</p>

Property	Editable?	Accepts Svar?	Description
Fault Selection			
JMS Property Name	Y	Y	JMS Property name used to send the fault as a value. Default property name is "faultName".

Interface Settings

Property	Editable?	Accepts Svar?	Description
Operation Selection			
Message Selector	Y	Y	A JMS message selector allows a client to specify, by message header and properties, the messages it's interested in. Message selector on Interface Settings is configurable when Error Action in Operation Selection is other than "Retain Message in Service Destination" and Operation Selection Type is "SCA".
Request Message			
Message Type	Y	Y	<p>Message type used for request messages. One of:</p> <ul style="list-style-type: none"> • XML Text - A text message carrying XML payload that confirms to specified schema. • Bytes - Binary data • Text - A text message carrying a payload of type xsd:string. • XML Bytes - XML content sent as bytes. (JMS resource instances treat this type as bytes but JMS bindings expect content in XML.) <p>Default: XML Text.</p>
Durable Subscription	Y	Y	<p>Configurable only in JMS Binding on Promoted Service.</p> <p>Specifies a durable subscription. You must specify a name in the Durable Subscription field which gets registered with the JMS application as the durable subscription name.</p> <div>  <p>Applicable only if Request Message Destination type is Topic.</p> </div>


Property	Editable?	Accepts Svar?	Description
Subscription Name	Y	Y	<p>Configurable only in JMS Binding on Promoted Service.</p> <p>The subscription name registered with the JMS application for durable subscriptions. This field is only available when the Durable Subscription field is checked.</p> <div>  <div>Applicable only if Request Message Destination type is Topic.</div> </div>
Delivery Mode	Y	Y	<p>Configurable only in JMS Binding on Promoted Reference.</p> <p>The delivery mode of the message. One of the following:</p> <ul style="list-style-type: none"> • Persistent - Messages are stored and forwarded • Non-Persistent - Messages are not stored and could be lost due to failures in transmission. • TIBCO Enterprise Message Service Reliable - The consumer never sends the provider a receipt confirmation or access denial and the provider does not wait for it. Reliable mode decreases the volume of message traffic, enabling higher message rates. <p>Default: Non-Persistent.</p>
Message Priority	Y	Y	<p>Configurable only in JMS Binding on Promoted Reference.</p> <p>Priority of the message. You can set the priority to a value from 0-9.</p>
Message Expiration	Y	Y	<p>Configurable only in JMS Binding on Promoted Reference.</p> <p>The time in milliseconds for which request message is retained by JMS Provider.</p>

Property	Editable?	Accepts Svar?	Description
Operation Timeout	Y	Y	<p>Configurable only in JMS Binding on Promoted Reference.</p> <p>The period that the JMS binding waits for the response to arrive.</p> <p>Default: If the MEP is In-Out, the defaults are 6000 ms at the port type and operation levels. If other values (non-default values) are specified, these values take effect, with the value at the operation level given precedence.</p> <div>  <p>Operation Timeout is applicable for a Reference only. For a Service, add a thread policy on a component service and set timeout on the thread policy.</p> </div>
Reply Message			
Message Type	Y	Y	<p>Message type used for reply messages. One of:</p> <ul style="list-style-type: none"> XML-Text - A text message carrying XML payload that confirms to specified schema. Bytes - Binary data Text - A text message carrying a payload of type xsd:string. xmlBytes - XML content sent as bytes. (JMS resource instances treat this type as bytes but JMS bindings expect content in XML.) <p>Default: XML-Text.</p>
Delivery Mode	Y	Y	<p>Configurable only in JMS Binding on Promoted Service.</p> <p>The delivery mode of the message. One of the following:</p> <ul style="list-style-type: none"> Persistent - Messages are stored and forwarded Non-Persistent - Messages are not stored and could be lost due to failures in transmission. TIBCO Enterprise Message Service Reliable - The consumer never sends the provider a receipt confirmation or access denial and the provider does not wait for it. Reliable mode decreases the volume of message traffic, enabling higher message rates. <p>Default: Non-Persistent.</p>


Property	Editable?	Accepts Svar?	Description
Message Priority	Y	Y	Configurable only in JMS Binding on Promoted Service. Priority of the message. You can set the priority to a value from 0-9.
Message Expiration	Y	Y	Configurable only in JMS Binding on Promoted Service. The time in milliseconds for which reply message are retained by JMS Provider.
Fault Message: This section is visible only in JMS Binding on Promoted Service and if operation has defined faults. It is applicable only for In-Out-Fault MEP.			
Override Reply Message	Y	N	Configuration from Reply Message is INHERITED by default. To "Override Reply Message" configuration in Interface Settings for Fault Message select "Override Reply Message".
Message Type	Y	Y	Message type used for reply messages. One of: <ul style="list-style-type: none"> XML-Text - A text message carrying XML payload that confirms to specified schema. Bytes - Binary data Text - A text message carrying a payload of type xsd:string. xmlBytes - XML content sent as bytes. (JMS resource instances treat this type as bytes but JMS bindings expect content in XML.) Default: XML-Text.
Delivery Mode	Y	Y	The delivery mode of the message. One of the following: <ul style="list-style-type: none"> Persistent - Messages are stored and forwarded Non-Persistent - Messages are not stored and could be lost due to failures in transmission. TIBCO Enterprise Message Service Reliable - The consumer never sends the provider a receipt confirmation or access denial and the provider does not wait for it. Reliable mode decreases the volume of message traffic, enabling higher message rates. Default: Non-Persistent.
Message Priority	Y	Y	Priority of the message. You can set the priority to a value from 0-9.


Property	Editable?	Accepts Svar?	Description
Message Expiration	Y	Y	The time in milliseconds for which reply message is retained by JMS Provider.

Operation Node

Property	Editable?	Accepts Svar?	Description
Operation Settings			
Name	N	N	Operation name.
Description	Y	N	Notes for operation name.
Operation Selection			
 Configurable only in JMS Binding on Promoted Service.			
Message Selector	Y	Y	A JMS message selector allows a client to specify, by message header, the messages it's interested in. Message Selector is displayed only when Operation Selection Type is "Custom" or Operation Selection Error Action is "Retain Message in Service Destination" and is used as a operation selector for the selected operation.
Request Message			
Override Request Message	Y	N	Override INHERITED Request Message configuration from Interface Settings for this operation only. If selected Message Type can be overridden.
Message Type	Y	Y	<p>Message type used for request messages. One of:</p> <ul style="list-style-type: none"> XML Text - A text message carrying XML payload that confirms to specified schema. Bytes - Binary data Text - A text message carrying a payload of type xsd:string. XML Bytes - XML content sent as bytes. (JMS resource instances treat this type as bytes but JMS bindings expect content in XML.) <p>Default: XML Text.</p>

Property	Editable?	Accepts Svar?	Description
Durable Subscription	Y	Y	Specifies a durable subscription. You must specify a name in the Durable Subscription field which gets registered with the JMS application as the durable subscription name. Durable subscription is displayed only when Request Message Destination Type is "Topic" and Operation Selection Type is "Custom" or Operation Selection Error Action is "Retain Message in Service Destination".
Subscription Name	Y	Y	The subscription name registered with the JMS application for durable subscriptions. This field is only available when the Durable field is checked and Request Message Destination Type is "Topic" and Operation Selection Type is "Custom" or Operation Selection Error Action is "Retain Message in Service Destination".
Delivery Mode	Y	Y	<p>Configurable only in JMS Binding on Promoted Reference.</p> <p>The delivery mode of the message. One of the following:</p> <ul style="list-style-type: none"> • Persistent - Messages are stored and forwarded • Non-Persistent - Messages are not stored and could be lost due to failures in transmission. • TIBCO Enterprise Message Service Reliable - The consumer never sends the provider a receipt confirmation or access denial and the provider does not wait for it. Reliable mode decreases the volume of message traffic, enabling higher message rates. <p>Default: Non-Persistent.</p>
Message Priority	Y	Y	<p>Configurable only in JMS Binding on Promoted Reference.</p> <p>Priority of the message. You can set the priority to a value from 0-9.</p>
Message Expiration	Y	Y	<p>Configurable only in JMS Binding on Promoted Reference.</p> <p>The time in milliseconds for which reply message are retained by JMS Provider.</p>

Property	Editable?	Accepts Svar?	Description
Operation Timeout	Y	Y	<p>Configurable only in JMS Binding on Promoted Reference.</p> <p>The period that the JMS binding waits for the response to arrive.</p> <p>Default: If the MEP is In-Out, the defaults are 6000 ms at the port type and operation levels. If other values (non-default values) are specified, these values take effect, with the value at the operation level given precedence.</p> <div>  <p>Operation Timeout is applicable for a Reference only. For a Service, add a thread policy on a component service and set timeout on the thread policy.</p> </div>
Reply Message			
Override Reply Message	Y	N	Override INHERITED Reply Message configuration from Interface Settings for this operation only.
Message Type	Y	Y	<p>Message type used for reply messages. One of:</p> <ul style="list-style-type: none"> XML Text - A text message carrying XML payload that confirms to specified schema. Bytes - Binary data Text - A text message carrying a payload of type xsd:string. XML Bytes - XML content sent as bytes. (JMS resource instances treat this type as bytes but JMS bindings expect content in XML.) <p>Default: XML Text.</p>

Property	Editable?	Accepts Svar?	Description
Delivery Mode	Y	Y	<p>Configurable only in JMS Binding on Promoted Service.</p> <p>The delivery mode of the message. One of the following:</p> <ul style="list-style-type: none"> • Persistent - Messages are stored and forwarded • Non-Persistent - Messages are not stored and could be lost due to failures in transmission. • TIBCO Enterprise Message Service Reliable - The consumer never sends the provider a receipt confirmation or access denial and the provider does not wait for it. Reliable mode decreases the volume of message traffic, enabling higher message rates. <p>Default: Non-Persistent.</p>
Message Priority	Y	Y	<p>Configurable only in JMS Binding on Promoted Service.</p> <p>Priority of the message. You can set the priority to a value from 0-9.</p>
Message Expiration	Y	Y	<p>Configurable only in JMS Binding on Promoted Reference.</p> <p>The time in milliseconds for which reply message are retained by JMS Provider.</p>
Fault Message  This section is visible only if faults are configured.			
Override Fault Message	Y	N	Override INHERITED fault message configuration from Interface Settings.
Fault Name	N	N	Name of the fault.
Message Type	Y	Y	<p>Message type used for reply messages. One of:</p> <ul style="list-style-type: none"> • XML Text - A text message carrying XML payload that confirms to specified schema. • Bytes - Binary data • Text - A text message carrying a payload of type xsd:string. • XML Bytes - XML content sent as bytes. (JMS resource instances treat this type as bytes but JMS bindings expect content in XML.) <p>Default: XML Text.</p>

Property	Editable?	Accepts Svar?	Description
Delivery Mode	Y	Y	<p>The delivery mode of the message. One of the following:</p> <ul style="list-style-type: none"> Persistent - Messages are stored and forwarded Non-Persistent - Messages are not stored and could be lost due to failures in transmission. TIBCO Enterprise Message Service Reliable - The consumer never sends the provider a receipt confirmation or access denial and the provider does not wait for it. Reliable mode decreases the volume of message traffic, enabling higher message rates. <p>Default: Non-Persistent.</p>
Message Priority	Y	Y	Priority of the message. You can set the priority to a value from 0-9.
Message Expiry	Y	Y	Configurable only in JMS Binding on Promoted Service. The time in milliseconds for which reply message are retained by JMS Provider.

Context Parameter Mapping

The following table shows the context parameter mapping to JMS header parameters or JMS application properties. JMS Header parameters or JMS application properties on JMS message from Request Message can be mapped to a context parameter and vice versa. Context parameters are defined in the General section of the Promoted Service or Promoted Reference. All the parameters defined in Context Parameters are available to Context Parameter Mapping in JMS Binding.

Property	Description
Context Parameter	Name of the context parameter.
Direction	<p>Direction of the flow of parameter.</p> <ul style="list-style-type: none"> INPUT: JMS Header parameter or JMS Application property is mapped to a Context Parameter. OUTPUT: Context parameter is mapped to a JMS Header parameter or JMS Application property.
Header Source	<p>Source of the header parameter.</p> <ul style="list-style-type: none"> JMS_HEADER: When JMS_HEADER is used, a JMS Header parameter name to map to a context parameter can be selected from Header Name. JMS_APPLICATION_PROPERTIES: Customer JMS Application property name is used for Context Parameter mapping.

Property	Description
Header Name	Shows JMS Header parameter names when JMS_Header is selected. Allows custom property value to be set when JMS_APPLICATION_PROPERTIES is set.



JMS Binding supports only Context Parameters of 'Type' Basic.


Context Parameters

A *context parameter* is a key-value pair passed to a service operation invocation. The values are populated by bindings, which map transport and binding headers to context parameters. Context parameters allow component developers to access transport and binding metadata that could affect the execution of an operation but which is not passed in the input, output, and fault messages defined by an abstract WSDL file.

A service may be supported on multiple types of transport bindings; each binding protocol specifies its own headers. For example, HTTP supports a body and headers that specify metadata that can be mapped to context parameters. The SOAP similarly defines a body and headers which are different than HTTP headers. The JMS protocol defines headers and allows developers to define application-specific properties. Typically, a client invoking a service will set some headers. For example, browsers usually set the HTTP Locale and Referrer headers.

Component implementations can read and set the values of context parameters and the values can be used to control service operation logic. The operation behavior thus changes according to the metadata. For example, consider a service that pays the agent that referred a customer to a website. To track the referrer on a SOAP/HTTP binding, you would specify a mapping from the HTTP Referrer header to a context parameter. If the service has a JMS binding, you would specify a mapping from a JMS message property named `referencedBy` to the same context parameter. When the incoming message is SOAP/HTTP, the HTTP Referrer header is copied into the context parameter and when a JMS message arrives, the `referencedBy` property is copied into the context parameter. The following table lists the header sources for each binding type.

Header Source

Binding Type	Header Source
SOAP/HTTP	HTTP Transport Header, HTTP Context, TCP Context, SOAP Header, SOAP Fault  The SOAP/HTTP reference binding interprets a response SOAP message as a Fault when the Body-element of the message has a Fault-element regardless of the HTTP status code.
SOAP/JMS	JMS Header, JMS Application Properties, SOAP Fault, SOAP Header
JMS	JMS Header, JMS Application Properties
REST	



In the TIBCO ActiveMatrix platform, the context parameter key `com.tibco.security.userinformation` is used to communicate security context. It can be retrieved by a component from `requestContext`. However, when invoking a reference this context parameter may be overwritten by a policy agent before the SOAP binding maps it to a HTTP Transport Header or JMS application property. Therefore, you cannot set this context parameter in a component before invoking a reference.

The following sections list the headers available in each header source. The tables in each section list which headers are available in service or reference bindings.

- From: XXX Binding To: Context applies to inbound messages received on either a service ("in" part of "in-out" MEP) or a reference ("out|fault" part of "in-out" MEP)
- From: Context To: XXX Binding applies to outbound messages sent from either a service ("out|fault" part of "in-out" MEP) or a reference ("in" part of "in-out" MEP)

HTTP Context

From: SOAP/HTTP Binding (WS-A = OFF)	To: Context
Service	HTTP-Method, HTTP-FileSpec, HTTP-Version
Reference	HTTP-Status-Code, HTTP-Status-Message

From: SOAP/HTTP Binding (WS-A = ON)	To: Context
Service	HTTP-Method, HTTP-FileSpec, HTTP-Version
Reference	None

TCP Context

From: SOAP/HTTP Binding	To: Context
Service	Local-TCP-Host, Local-TCP-Port, Remote-TCP-Host, Remote-TCP-Port
Reference	None

SOAP Fault

From: SOAP Binding (For Declared Faults)	To: Context
Service	None
Reference	Role, Code

From: Context	To: SOAP Binding (For Declared Faults)
Service	Role, Code
Reference	None

JMS Header







From: SOAP/JMS Binding	To: Context
Service	JMSCorrelationID, JMSDeliveryMode, JMSMessageID, JMSType
Reference	None

From: Context	To: SOAP/JMS Binding
Service	None
Reference	JMSCorrelationID, JMSDeliveryMode, JMSType

Creating Context Parameters

You can create context parameters in the Properties view. As part of parameter creation, you specify the WSDL operation.

Procedure

1. Click a service or reference.
2. In the Properties view, expand the **Context Parameters** node.
3. Click the plus icon ().
A parameter named contextParameter*n* is added to the Context Parameters table. By default the parameter is available to the Input direction of all the operations available in the port type and the parameter type is set to basic with definition string. A error badge is added to components wired to service or reference.
4. Click the **Name** column and type a name for the parameter.
5. To change the selected operations, click the **Operations** column and click .
The WSDL Operations dialog displays.
6. Uncheck the checkboxes next to the operations to which the parameter will not apply and click **OK**.
7. To change the message direction to which the parameter applies, click the **Direction** column and click .
The Select Direction dialog displays.
8. Choose Input, Output, or Fault. If the message type is Fault, choose the fault messages to which the parameter applies. Click **OK**.
9. Click the Type column, click the , and choose the type of the parameter.
10. Click the Definition column and choose the parameter definition:
 - Basic - Click the  and choose a definition.
 - Message - Click the . The Select Message and Message Part dialog displays. Choose the message and message part to which the parameter applies and click **OK**.
11. Click **Save**.
An error badge is added to the service or reference.
12. Right-click the service or reference and select **Quick Fixes > Update Context Mapping Parameters**.
The parameter is added to the Context Parameter Mapping table of the service's or reference's bindings.
13. If the service or reference is wired to a component, update the component implementation by right-clicking the component and selecting **Quick Fixes > Update Implementation Class**.
The implementation will be updated to enable access to the context parameter.

Mapping Context Parameters to Header Fields

Procedure

1. Click the service or reference.
2. In the Properties view, click the **Binding** tab.
3. Expand a binding node.
4. Click **Context Parameter Mapping**.
5. In the Context Parameter Mapping table, click a context parameter.
6. Click the **Header Source** and **Header Name** columns and configure the mapping.

Context Parameter Reference

Context Parameters

Column	Description
Name	The name of the context parameter.
Operations	The operations in the port type of the service or reference to which the parameter applies.
Direction	The direction of the message to which the parameter applies: Input, Output, Fault. If Fault, the list of fault messages.
Type	<p>The type of the parameter:</p> <ul style="list-style-type: none"> • Basic - a basic property. • Message - a message associated with the binding. The binding cannot reference a message used by an operation in the port type. • Bag - a collection of properties.
Definition	<ul style="list-style-type: none"> • Basic - the definition of the property. If the Direction is Input or Output, one of string, int, long, or boolean. If the Direction is Fault, one of string, int, long, boolean, QName, or URI. • Message - the definition of the message, consisting of a message and a message part.

Enabling Transactions in SOAP and JMS Bindings

You can enable transactions in a SOAP or JMS binding by configuring intents and policy sets.

Prerequisites

All components must be enabled with the Global Managed Transaction intent.

For information on intents and policy sets, refer to [Policy Management](#).

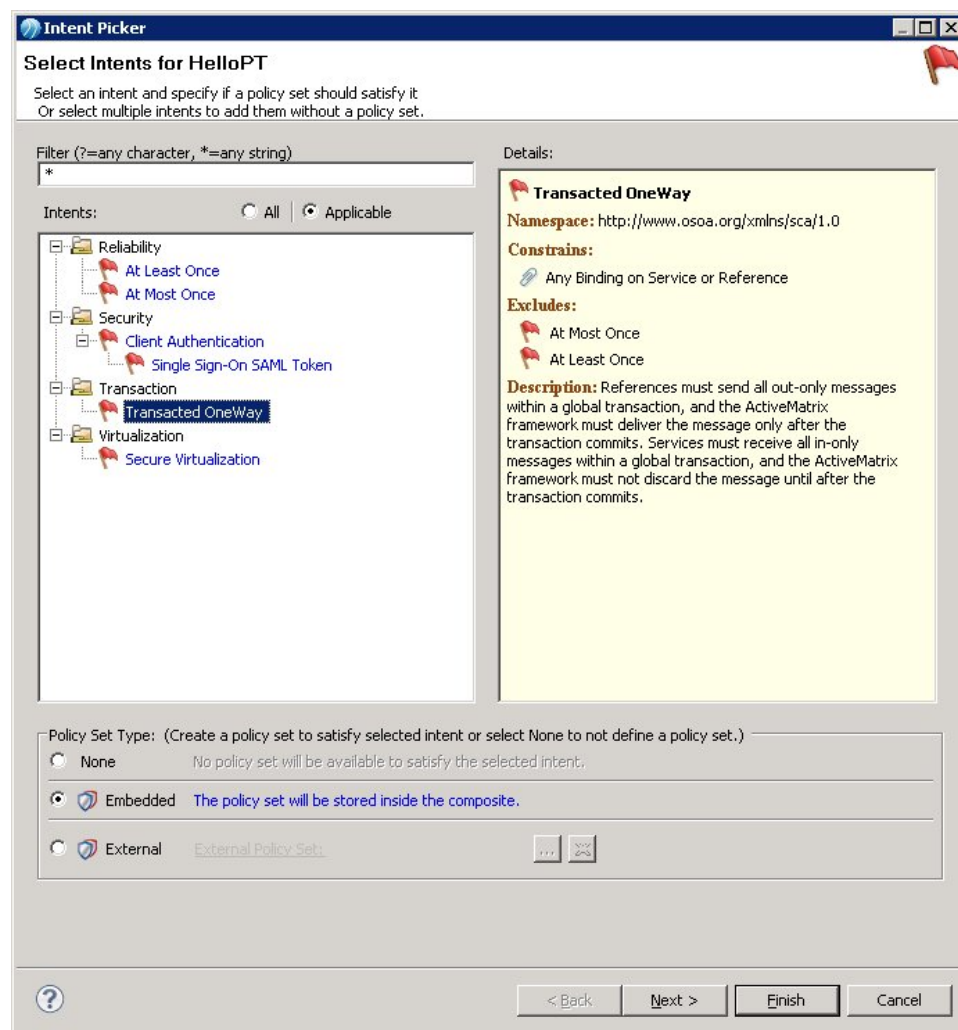
The following limitations apply when using transactions with SOAP and JMS bindings:

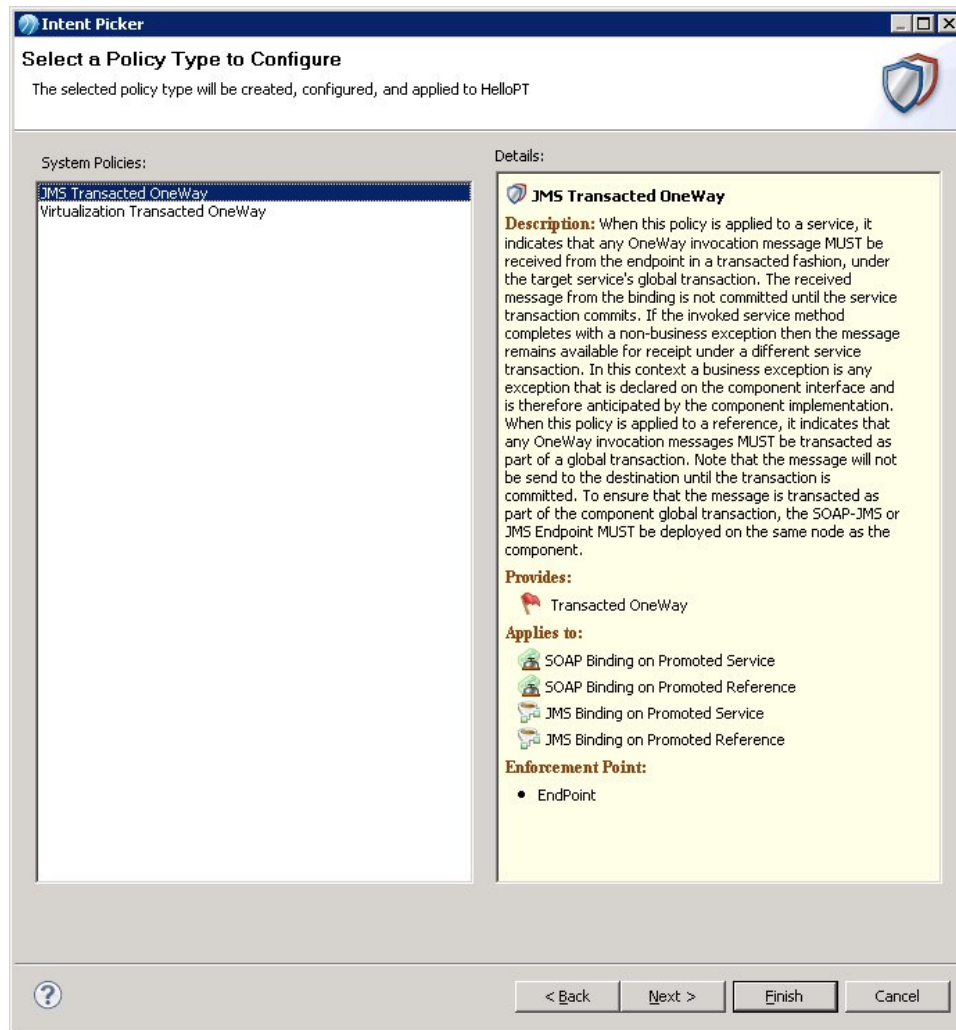
- Transactions are supported only in a single node.
- Transactions are supported only for In-only MEP.
- Configuring Transacted OneWay intent and JMS Transacted OneWay policy set in the Administrator UI is not supported.

Here is a summary of the key steps for enabling transactions in SOAP or JMS bindings (both on the service and reference side):

Procedure

1. For the SOAP or JMS binding, select **Transacted OneWay** as the intent and **JMS Transacted OneWay** as the policy set.





2. For all components, select **Global Managed Transactions** as the intent and **ManagedTransaction** as the policy set. This is a pre-requisite for JMS Transacted OneWay.
3. Ensure that the JDBC resource template is configured correctly:
 - a) The Connection Type field is set to XA.
 - b) The right data source class is set in the Data Source Class Name field.
4. Ensure that the JMS Connection Factory resource template is configured correctly. The Connection Factory field is set to XAConnectionFactory.
5. For the Enterprise Message Service server used by the SOAP or JMS binding, ensure that XAConnectionFactory is present in the connection factory definitions section of the factories.conf file.

```
[XAConnectionFactory]
type = xageneric
url = tcp://port
```

The default location of the factories.conf file depends on the operating system:

- Windows - C:/Documents and Settings/user name/Application Data/EMS_HOME/tibco/cfgmgmt/ems/data/factories.conf.
- Unix - EMS_HOME/tibco/cfgmgmt/ems/data/factories.conf.

Policy Management

Policy management involves specifying a capability or constraint on a governed object — Composite, Component, Service, or Reference — to affect runtime behavior such as security, reliability, transactions, threading, and quality of service.

Constraints and capabilities are managed separately from the core business logic of your Application. To enable policy management, an application designer specifies intents and policy sets.

Policies specified at design time are packaged into the deployment archive and enforced via a governance agent embedded in the TIBCO ActiveMatrix runtime.

An *intent* describes abstract constraints on the behavior of a Component, or on interactions between Components. Intents let application designers specify requirements in a high-level, abstract form, independent of the configuration details of the runtime and bindings. Intents guide administrators as they configure bindings, policies, and runtime details.

A *policy* is a configuration that specifies the details that TIBCO ActiveMatrix needs to enforce a constraint declared in an intent. A policy can also be specified without an intent.

A *policy set* contains one or more policies. Adding a policy set to a governed object applies its Policies at the object.

TIBCO ActiveMatrix support for intents, policy sets, and policies conforms to the [SCA Policy](#) specification. TIBCO ActiveMatrix also uses TIBCO ActiveMatrix Policy Director Governance, which is a governance solution to manage and enforce cross-functional requirements such as security, monitoring, and compliance independent of their implementation and deployment. For more details, see "TIBCO ActiveMatrix Policy Director Governance" in *TIBCO ActiveMatrix BPM SOA Concepts*.

Policies

A *policy* is a configuration that specifies the details that TIBCO ActiveMatrix needs to enforce a constraint declared in an intent. A policy can also be specified without an intent.

Active Matrix supports many types of policies, such as authentication, authorization, encryption, transaction, reliability, and threading. For example, an authentication policy at a service with a SOAP binding enforces authentication of all SOAP requests from clients. Though the business logic in your application may require a policy to enforce authentication, the business logic and the policy operate independently. A managed transaction policy ensures that a managed transaction is started a composite or component runs, so its operations execute in a transaction environment.

Governed Objects

A *governed object* is an element at which TIBCO ActiveMatrix can enforce policies.

The following elements can be governed objects:

- Composite
- Component
- Component service or reference
- Promoted service or reference
- Binding

Intents

An *intent* describes abstract constraints on the behavior of a component, or on interactions between components. Intents let application designers specify requirements in a high-level, abstract form, independent of the configuration details of the runtime and bindings. Intents guide administrators as they configure bindings, policies, and runtime details.

Intents apply at specific types of services, references, bindings, and components. If you specify an intent on a composite, TIBCO Business Studio applies it at all the appropriate services and references within the composite but not in sub-composites.





Security intents can only be satisfied by external policy sets.

Configuring Intents



You can configure intents for a composite, a service, a reference, or a binding object.

Procedure

1. Click an element and follow the procedure.

Element	Procedure
Composite, Service, Reference	<p>Right-click the element or start in the Properties view.</p> <ul style="list-style-type: none"> • Right-click the composite , service, or reference and select Add > Intent... • 1. In the Properties view, click the Policies tab. 2. Click .
Binding	<ol style="list-style-type: none"> 1. In the Properties view, click the Bindings tab. 2. Click a binding. 3. Click .

2. In the Intent Picker, click one or more intents.
3. In the Policy Set Type area, choose the type of policy set to satisfy the intents and follow the appropriate procedure.

Policy Set Type	Procedure
None	<ul style="list-style-type: none"> • One intent <ol style="list-style-type: none"> 1. Click the None radio button. • Multiple intents <ol style="list-style-type: none"> 1. None radio button selected automatically. <p> If you do not select a policy set an error badge is added to each unsatisfied intent. To resolve the error, click the intent and then click fix... in the right pane.</p>
Embedded (one intent)	<ol style="list-style-type: none"> 1. Click the Embedded radio button.
External (one intent)	<ol style="list-style-type: none"> 1. Click the External radio button. 2. Click the ellipsis image (). The Policy Set picker displays. <ul style="list-style-type: none"> • Click a policy set.

Policy Set Type	Procedure
	<ul style="list-style-type: none"> Click Create Policy Set. <ol style="list-style-type: none"> Accept the folder to contain the policy set or select a new one. Enter the policy set file name, target namespace, and default policy set name. Click Finish.

- Click **Next**.
- In System Policy List, click a policy to satisfy the intent and click **Next**.
- In the policy's properties display, configure the policy properties and click **Finish**.

Intents Reference

Category	Intent
Life Cycle	Start Services First Intents - When a component reference is wired to a component service, the provider component must start before the referencing component starts. Use this intent when a consumer might reference the provider soon after it starts.
	Prepare Before Undeploy Intents - This component requires preparation before undeploying. Before Administrator can begin to undeploy the component, the component must explicitly confirm to Administrator that preparations are complete.
Reliability	At Least Once Intents - The provider must receive every message sent to it by consumers at least once. It may receive some messages more than once.
	At Most Once Intents - The provider must not receive any message more than once. Even if the provider fails to process a message, the Messaging Bus must not re-deliver it. It is acceptable that the provider might not receive some messages.
Security	Authorization Intents - Promoted services must authorize consumers.
	Client Authentication Intents - Services must authenticate the consumer's identity.
	Credential Mapping Intents - References (and promoted services) must propagate the consumer's identity to providers.
	Consumer Confidentiality Intents - Ensure consumer-side confidentiality with encryption when forwarding and decryption when receiving.
	Consumer Integrity Intents - Promoted references with SOAP bindings must send only signed requests.

Category	Intent
	Provider Confidentiality Intents - Promoted services with SOAP bindings must accept only encrypted requests, and decrypt them. Optionally, they may also encrypt responses.
	Provider Integrity Intents - Promoted services with SOAP bindings must accept only signed requests, and verify the signature.
Transaction	Managed Transaction Intents - Providers require an open transaction for each request message.
	Transacted OneWay Intents - References must send all out-only messages within a global transaction, and TIBCO ActiveMatrix delivers the message only after the transaction commits.
Virtualization	Secure Virtualization Intents - References must send all messages to the Messaging Bus using SSL. Services must receive all messages from the Messaging Bus using SSL.

At Least Once Intents

Designers can specify an at least once intent to require that providers receive every message at least once.

Reliability intents usually apply at service virtualization bindings. If you apply them to composites, service virtualization bindings of the composite inherit them, as appropriate.

Category

Reliability

Qualifier	Description
<i>none</i>	<p>The provider must receive every message sent to it by consumers at least once. It may receive some messages more than once.</p> <p>If the node or the Message Bus fails, it must deliver messages after it restarts. Messages must be persistent; if the provider cannot successfully process a message, the Message Bus must re-deliver it after a delay. The consumer must send every message using the persistent mode of the Messaging Bus.</p>

At Most Once Intents

Designers can specify an at most once intent to require that providers receive every message at least once.

Reliability intents usually apply at service virtualization bindings. If you apply them to composites, service virtualization bindings of the composite inherit them, as appropriate.

Category

Reliability

Qualifier	Description
<i>none</i>	The provider must not receive any message more than once. Even if the provider fails to process a message, the Messaging Bus must not re-deliver it. It is acceptable that the provider might not receive some messages.

Authorization Intents

Designers can specify an authorization intent to require that services must authorize the consumer before processing a request.

In actual practice, authentication and authorization requirements often apply in tandem. Policies that satisfy client authentication intents often provide authorization information as a side effect.

Authorization intents usually apply at services. If you apply them to composites, services of the composite inherit them, as appropriate.

Category

Security

Qualifier	Description
<i>none</i>	Promoted services must authorize consumers. Prior authentication must supply authorization information.
Role (default qualifier)	Promoted services must authorize consumers based on role information. Prior authentication must supply authorization information.

Client Authentication Intents

Designers can specify a client authentication intent to require that services must authenticate the consumer's identity before processing a request.

In actual practice, authentication and authorization requirements often apply in tandem. Policies that satisfy client authentication intents often provide authorization information as a side effect.

Client authentication intents usually apply at services. If you apply them to composites, services of the composite inherit them, as appropriate.

Category

Security

Qualifier	Description
<i>none</i>	Services must authenticate the consumer's identity.
Basic	Promoted services must authenticate the consumer's identity using HTTP basic authentication.
Username Token (default qualifier)	Promoted services must authenticate the consumer's identity using a username token in the SOAP WS Security header.

Qualifier	Description
Single Sign-On SAML Token	<p>Component services and promoted references must authenticate the consumer's identity using a single sign-on SAML token.</p> <p>Authentication policies generate the SAML token; Single Sign-On SAML Credential Mapping policies propagate the token at promoted references.</p> <p>Single Sign-On SAML Token is the only qualifier that can be specified for a Virtualization binding.</p>
WS Security SAML Token Profile	Promoted services with SOAP bindings must authenticate the consumer's identity using a SAML 1.1 or SAML 2.0 assertion in the SOAP WS Security header.
X509	Promoted services with SOAP bindings must authenticate the consumer's identity using the consumer's X509 signature.

Consumer Confidentiality Intents

Designers can specify an consumer confidentiality intent to require consumer-side confidentiality with encryption when forwarding and decryption when receiving.

Consumer confidentiality intents apply only at promoted SOAP references. If you apply them to composites, references of the composite inherit them, as appropriate.

Category

Security

Qualifier	Description
<i>none</i>	Ensure consumer-side confidentiality with encryption when forwarding and decryption when receiving.
WS Security (default qualifier)	Ensure consumer-side confidentiality with WS Security encryption when forwarding and decryption when receiving.

Consumer Integrity Intents

Designers can specify an consumer integrity intent to require that promoted references with SOAP bindings must send only signed requests.

Consumer integrity intents apply only at promoted references with SOAP bindings. If you apply them to composites, references of the composite inherit them, as appropriate.

Category

Security

Qualifier	Description
<i>none</i>	<p>Promoted references with SOAP bindings must send only signed requests.</p> <p>Optionally, they may also accept signed responses and verify the signature.</p>

Qualifier	Description
WS Security (default qualifier)	Promoted references with SOAP bindings must send only signed requests. Optionally, they may also accept signed responses and verify the signature. Signature information travels in the WS Security header.

Credential Mapping Intents

Designers can specify a credential mapping intent to require that references must propagate the consumer's identity to providers.

Credential mapping intents usually apply at references.

Some qualifiers also apply at promoted services. After the service enforces an authentication policy, it must propagate the resulting identity information.

If you apply credential mapping intents to composites, the references and services of the composite inherit them, as appropriate.

Category

Security

Qualifier	Description
<i>none</i>	References (and promoted services) must propagate the consumer's identity to providers.
Single Sign-On SAML Token (default qualifier)	References (and promoted services) must propagate a SAML token asserting the consumer's identity to providers within the TIBCO ActiveMatrix environment. Single Sign-On SAML Token is the only qualifier that can be specified for a Virtualization binding.
WS Security SAML Profile	References (and promoted services) must propagate a SAML token asserting the consumer's identity; promoted references must insert the token in the WS Security header.
Username Token	Promoted references must insert a username token representing the consumer's identity in the WS Security header.
Basic	Promoted references must insert a username token representing the consumer's identity in the HTTP basic authentication header.

Managed Transaction Intents

Designers can specify a managed transaction intent to require that messages must be part of a transaction.

Managed transaction intents apply at component implementations.

When you add this intent, TIBCO Business Studio automatically embeds an appropriate policy set that satisfies the intent.

Category

Transaction

Qualifier	Description
<i>none</i>	Providers require an open transaction for each request message.
Global (default qualifier)	Providers require an open global transaction for each request message. If an open global transaction is not propagated from a consumer, then TIBCO ActiveMatrix starts one before delivering the message to the provider; furthermore, TIBCO ActiveMatrix either commits that transaction when the provider sends its response message, or rolls back that transaction when the provider sends a fault message.

Prepare Before Undeploy Intents

Designers can specify a prepare before undeploy intent to declare that a component requires preparation before undeploying. Administrator must not undeploy until the component explicitly confirms that preparations are complete.

Prepare before undeploy intents apply to components.

Category

LifeCycle

Qualifier	Description
<i>none</i>	This component requires preparation before undeploying. Before Administrator can begin to undeploy the component, the component must explicitly confirm to Administrator that preparations are complete.

Provider Confidentiality Intents

Designers can specify an provider confidentiality intent to require that promoted services with SOAP bindings must accept only encrypted requests, and decrypt them.

Provider confidentiality intents apply only at promoted services with SOAP bindings. If you apply them to composites, services of the composite inherit them, as appropriate.

Category

Security

Qualifier	Description
<i>none</i>	Promoted services with SOAP bindings must accept only encrypted requests, and decrypt them. Optionally, they may also encrypt responses.
WS Security (default qualifier)	Promoted services with SOAP bindings must accept only encrypted requests, and decrypt them. Optionally, they may also encrypt responses. Encryption information travels in the WS Security header.

Provider Integrity Intents

Designers can specify an provider integrity intent to require that promoted services with SOAP bindings must accept only signed requests, and verify the signature.

Provider integrity intents apply only at promoted services with SOAP bindings. If you apply them to composites, services of the composite inherit them, as appropriate.

Category

Security

Qualifier	Description
<i>none</i>	Promoted services with SOAP bindings must accept only signed requests, and verify the signature. Optionally, they may also sign responses.
WS Security (default qualifier)	Promoted services with SOAP bindings must accept only signed requests, and verify the signature. Optionally, they may also sign responses. Signature information travels in the WS Security header.

Secure Virtualization Intents

Designers can specify a secure virtualization intent to require SSL transport security for all messages.

Client authentication intents usually apply at service virtualization bindings. If you apply them to composites, service virtualization bindings of the composite inherit them, as appropriate.

Category

Virtualization

Qualifier	Description
<i>none</i>	References must send all messages to the Messaging Bus using SSL. Services must receive all messages from the Messaging Bus using SSL.

Start Services First Intents

Designers can specify a client start services first intent to require that before a referencing component can start, it must first wait until the provider component has started.

Start services first intents usually apply at reference that are wired to services. If you apply them to composites, wired references of the composite inherit them, as appropriate.

Administrator enforces the start services first intent only when the reference and service are distributed on the same node. If the target service of a reference is running on a different node, there is no guarantee that the target service will be running before the component is started.

Category

LifeCycle

Qualifier	Description
<i>none</i>	When a component reference is wired to a component service, the provider component must start before the referencing component starts. Use this intent when a consumer might reference the provider soon after it starts.

Transacted OneWay Intents

Designers can specify a transacted OneWay intent to require that one-way messages be part of a global transaction.

Transacted OneWay intents apply at Service Virtualization bindings. If you apply them to composites, bindings of the composite inherit them, as appropriate.

When you add this intent, TIBCO Business Studio automatically embeds an appropriate policy set that satisfies the intent.

Category

Transaction

Qualifier	Description
<i>none</i>	<p>References must send all out-only messages within a global transaction, and TIBCO ActiveMatrix delivers the message only after the transaction commits.</p> <p>Services must receive all in-only messages within a global transaction, and TIBCO ActiveMatrix does not discard the message until after the transaction commits.</p>

Policy Sets

A *policy set* contains one or more policies. Adding a policy set to a governed object applies its policies at the object. Policy sets are atomic transactional units for policy administration. When you add a policy set, you are explicitly requesting that TIBCO ActiveMatrix enforce *all* its policies for the governed object.

Policy sets can either be *embedded* in a composite file, or *external* (that is, specified in a separate XML file).

Policy Set Enforcement

- **Component service:** all bindings on all promoted services to which the policy applies inherit the policy. In inherited policy sets, the virtualization and HelloWorld1SOAP bindings inherit the Threading policy from the HelloWorldPT service JavaHelloComponent.
- **Promoted service:** all bindings on the service to which the policy applies inherit the policy. In inherited policy sets, the HelloWorld1SOAP binding inherits the WSAddressing policy applied to the service HelloWorldPT.
- **Binding:** if the policy applies to a binding it will be enforced. However, if you add a policy applicable to Virtualization binding to a SOAP binding, it takes effect on the virtualization proxy between the SOAP binding and component. In , the Virtualize policy applied to HelloWorld1SOAP binding takes effect on the service virtualization proxy (SV-Proxy).

Embedded Policy Sets

When you configure policy parameters using TIBCO Business Studio, TIBCO Business Studio automatically translates them into an embedded policy set, and attaches it to the composite. An embedded policy set generally configures exactly one policy.

The validations in TIBCO BusinessStudio prevents an invalid PolicySet configuration related to Global Managed Transaction policy.

External Policy Sets

When you configure an external policy set, it can define several policies that all apply to the same set of element types.

The XML `provides` and `appliesTo` properties pertain to external policy sets.



Security intents can only be satisfied by external policy sets.

provides

The `provides` property specifies the intents that the policy set satisfies. Its value is a list of intents, separated by the space character.

When you add a policy set, TIBCO Business Studio uses this information to determine which intents the policy set satisfies, and which intents still require policies.

appliesTo

The `appliesTo` property specifies specific binding types or specific implementation types which are the targets of the policy set. Its value is a list of types, separated by the vertical bar character (`|`).



When you add a policy set, TIBCO Business Studio applies its policies to all the elements (within that scope) that match any of the types in the `appliesTo` property. The resulting set of elements constitutes the governed object group of the policy set.

Displaying the Policy Sets Picker

You can display a Policy Sets Picker for a composite, a service, a reference, or a binding object.

Procedure

- Click an element and follow the procedure.


Element	Procedure
Composite, Service, Reference	<p>User either the right-button menu or the Properties view.</p> <ul style="list-style-type: none"> Right-click the element and select Add > Policy Set... In the Properties view: <ol style="list-style-type: none"> Click the Policies tab. Click .
Binding	<ol style="list-style-type: none"> In the Properties view, click the Bindings tab. Click a binding. Click .

Configuring Policy Sets

You can configure policy sets from the policy set picker. The procedure differs for embedded policy sets or external policy sets.

Procedure

1. Click a composite object and display its policy set picker.
2. In the Policy Set Type area, choose the type of policy set and follow the appropriate procedure.

Policy Set Type	Procedure
Embedded	<ol style="list-style-type: none"> 1. Click the Embedded radio button. 2. In System Policy List, click one or more policies. 3. If you select one policy and the Next > button is enabled, click Next > and configure the policy properties. 4. Click Finish.
External	<ol style="list-style-type: none"> 1. Click the External radio button. 2. Click . The Select Policy Set picker displays. 3. If no policy sets are listed, click Create Policy Set. <ol style="list-style-type: none"> a. Accept the folder to contain the policy set or select a new one. b. Enter the policy set file name, target namespace, and default policy set name. c. Click Finish. 4. Click Finish.

Editing Embedded Policy Sets

You can edit embedded policy sets from the policy set picker.

Procedure


1. Click a composite object and display its policy set picker.
2. Click a policy set.
The policy set details display in the pane on the right.
3. In the Operations area, click **Edit**.
The policy set's properties editor displays.
4. Edit the properties.
5. Click **Finish**.

Creating an External Policy Set

The process for creating external policy sets differs for resource templates or for composite, service, reference, and binding objects.

Procedure

1. Choose a starting point and follow the appropriate procedure.

Starting Point	Procedure
Resource Templates	1. Right-click the Resource Templates folder of a TIBCO SOA Project and select New... > PolicySets Resource Template .
Composite, Service, Reference, Binding	1. Display the element's policy set picker. 2. In the Policy Set Type radio button, click External . 3. In the External Policy Set field, click the External Policy Set link or  .

The Select Policy Set Resource dialog displays.

2. Click **Create Policy Set**.
The Create Policy Set Resource dialog displays.
3. Enter the policy set file name, target namespace, and default policy set name and click **Finish**.

Configuring External Policy Sets with XML

You can configure policy sets by editing a policy set resource file. Samples are available in the *TIBCO_HOME* directory.

Procedure

1. In the composite editor, click a service, a reference, a composite, or a component.
2. In the Properties view, select the **Policies** tab.
3. In the tree view, click an intent.
The right pane displays details of the intent. If the intent is not yet satisfied, TIBCO Business Studio reports this situation as a problem.
4. Click the **fix...** link, and select the fix **Attach Policy Set**.
5. Click the **Create Policy Set** button.
A dialog opens to create a new policy set resource.
6. Determine the resource location, filename, target namespace and policy set name and click **Finish**.
Store policy sets in a separate folder under the top level of project directory.
TIBCO Business Studio adds an empty policy set to the tree, and marks the intent as *Satisfied by* the new policy set.
7. Click the **Satisfied by** link in the intent.
TIBCO Business Studio opens a skeleton policy set resource file for editing. The filename extension is *.policysets*
8. Copy a policy sample, and paste it within the `<sca:policyset>` tag. For samples, see [Policy Templates Reference](#).
9. Modify the parameters of the sample, as needed.

- Each `.policysets` file contains one `sca:PolicySet`, which contains one top-level `.`
 - The `wsp:Policy` may contain nested `wsp:Policy` elements.
 - The top-level `wsp:Policy` element must specify the `template` attribute, and the namespace of the template.
10. Modify the skeleton, if needed.
- The skeleton begins with a list of namespaces. When you add policies that require namespaces that are not included in the skeleton, specify those namespace declarations in the top-level `wsp:Policy` element.
 - The skeleton also includes an `sca:policySet` element, which is empty except for its three attributes (`name`, `provides` and `appliesTo`).
 - To expand a policy set to provide several intents, add intents, separated by space characters, to the `provides` attribute.
 - To restrict a policy set to apply to a narrower set of components, you may remove components from the `appliesTo` attribute (separating components with vertical bar characters).
11. Save the policy set file.

Embedded Policy Sets Reference

With the exception of the Managed Transaction policy, you can configure each policy with the GUI. For policies that require external policy sets, see [Policy Templates Reference](#).

- [JMS At Most Once](#) - The JMS At Most Once policy ensures that the Messaging Bus attempts to deliver each message to its destination at most once.
- [JMS At Least Once](#) - It ensures that the JMS messages are delivered at least once to the recipients in case of exception scenarios by using the redelivery mechanism.
- [JMS Transacted OneWay](#) - It ensures delivery of one-way messages by wrapping message operations (send or receive) within transactions.
- [Message Transmission Optimization Mechanism \(MTOM\)](#) - A MTOM policy enables you to configure how binary data (attachments such as images, PDF files, and so on) is to be transmitted over SOAP.
- [Secure Virtualization](#) - Ensures that a service receives all messages from the Messaging Bus using SSL.
- [Threading](#) - Ensures that message dispatch and send operations use a thread pool.
- [Virtualization At Least Once](#) - Ensures that the Messaging Bus attempts to deliver each message to its destination at least once.
- [Virtualization Transacted OneWay](#) - Ensures that delivery of one-way messages are included as part of a transaction.
- [Virtualize](#) - Ensures that messages travel through the Messaging Bus, blocking potential optimization, while enabling beneficial side effects of the Messaging Bus.
- [Managed Transaction](#) - A Managed Transaction policy starts a managed transaction before invoking a component service, so its service implementation executes in a transaction environment.
- [Prepare Before Undeploy](#) - A Prepare Before Undeploy policy ensures that Administrator does not undeploy an application or component until the application or component explicitly confirms that its preparations are complete.
- [Start Service First](#) - Ensures that when a reference is wired to a service, the provider component starts before the referencing component starts.
- [WS-Addressing for References](#) - A WS-Addressing for Reference policy set ensures that a reference sends messages with WS-Addressing semantics.

- [WS-Addressing for Services](#) - A WS-Addressing for Services policy ensures that a service receives messages with WS-Addressing semantics.
- [WS-Reliable Messaging](#) - A WS-Reliable Messaging policy ensures that a service receives messages or a reference sends messages with WS-Reliable Messaging semantics.

JMS At Most Once

The JMS At Most Once policy ensures that the Messaging Bus attempts to deliver each message to its destination at most once.

If the delivery attempt fails for any reason, the Messaging Bus does not attempt to redeliver it. Instead, it removes the message from the queue immediately. It is permissible that the provider never successfully receives the message.

This policy set can apply at a Virtualization binding on a service or reference, or at a SOAP binding on a promoted service or reference.

JMS At Least Once

The JMS At Least Once policy applies to SOAP and JMS Bindings. It ensures that the JMS messages are delivered at least once to the recipients in case of exception scenarios by using the redelivery mechanism.

JMS AtLeastOnce policy is supported only when the configured JMS provider is Enterprise Message Service or IBM MQ.

The JMS AtLeastOnce policy can be applied on JMS Binding on promoted service as well as promoted reference. At JMS Service Binding, policy will ensure that no message loss happens in case of exception scenarios and at JMS Reference Binding, policy dictates that the JMS Message delivery mode is "PERSISTENT".

Limitations

- JMS At Least Once policy can only be used to satisfy At Least Once intent for the JMS transport bindings - JMS and SOAP/JMS bindings.
- JMS At Least Once policy supports only In-only MEP.
- JMS At LeastOnce policy can coexist with Virtualization At Least Once, Virtualize, Secure Virtualization and Threading Policy. It cannot coexist with JMS Transacted One Way, Virtualization Transacted OneWay, or At Most Once policies at any enforcement point.
- Only TIBCO EMS and IBM MQ JMS providers are supported. An error is displayed if any other JMS provider is configured. If JNDI initial context factory is configured for any JMS provider other than TIBCO EMS or MQ, an error message is displayed.
- If an error queue has not been created and the JMS Provider does not support dynamic queue creation or the user of the provider does not have create permissions, JMS binding deployment fails. If the error queue is deleted after the application is deployed, JMS binding attempts to move the erroneous message to the error queue. On failing to do so, it indefinitely attempts to move the message to the error queue regardless of the redelivery attempt configuration.

Properties

Following properties are configurable on JMS AtLeastOnce policy configured on JMS Binding on Promoted Service.

The JMS At Least Once policy configured on JMS Binding on Promoted Reference only indicates that the message delivery mode is "PERSISTENT" and no redelivery configuration is allowed.

Parameter	Description
Max redelivery count	<p>The JMS binding continues to attempt delivery until either delivery succeeds, or the number of delivery attempts reaches this value. This value specifies the total number of times the message is delivered, including the first delivery.</p> <p>For example:</p> <ul style="list-style-type: none"> • If the value is 1, the Messaging Bus attempts only the initial delivery and then stops. • If the value is 3, the sequence is as follows: <ol style="list-style-type: none"> 1. The message is delivered once. (first attempt) 2. On failure, the message is delivered again. (second attempt) 3. On failure, the message is delivered again. (third attempt) 4. A failure is reported. <p>Default: 0.</p>
Redelivery delay interval (s)	<p>After each failed delivery attempt, the JMS binding waits for this interval before its next attempt.</p> <p>Default: 15.</p>
Exceptions upon which to stop redelivery	<p>If the provider throws any exception in this list, then the JMS binding cancels further delivery attempts (even if the maximum redelivery count has not yet reached).</p> <p>Default: None.</p>
Enable error queue	<p>Indicate how a message is treated when delivery fails—that is, the JMS binding exceeds the maximum delivery count or the service implementation throws an error that stops delivery.</p> <p>When checked, place the message on the error queue defined in the Error queue name field. When unchecked, discard the message.</p> <p>Default: Unchecked.</p>
Transactional	<p>Indicates whether move to error queue should happen in the same session transaction in which message from source destination was received.</p> <div data-bbox="751 1654 798 1705"> </div> <div data-bbox="852 1654 1442 1717"> <p>Transaction here is not an XA transaction, this is only a session transaction.</p> </div>

Parameter	Description
Error queue name	<p>The queue on which JMS binding places failed message if the error queue is enabled.</p> <p>A warning is displayed on the Promoted Service indicating configured error queue should be present on the JMS provider.</p> <p>Default: Default error queue name is JMSAtLeastOnce_ErrorQueue.</p>

The JMS AtLeastOnce policy can be configured with the redelivery configuration by setting redelivery count and redelivery delay. In case when delivery fails, message is redelivered as per the configured redelivery count and with the redelivery delay. When message is moved to error queue following additional string properties describing the cause are added to the JMS message.

Property	Description
_JMSJCA_ExceptionMessage	Message from exception stack trace i.e. cause of redelivery.
_JMSJCA_ExceptionClass	Java exception class
_JMSJCA_ExceptionStackTrace	Stack trace of the exception occurred which initiated redelivery.
DeliveryFailureReason	Failure reason if other than exception message otherwise exception message.
AMX_NODE_NAME	Name of the AMX NODE on which application is deployed.
AMX_HOST_NAME	Name of the AMX HOST, which contains AMX Node.
AMX_ERROR_MSG	Property indicating whether this message is error message, TRUE when message is moved to error queue.
SourceDestinationName	Name of the Source destination from which message was received.
SourceDestinationType	Destination type of the source destination.

JMS Transacted OneWay

The JMS Transacted OneWay policy applies to SOAP and JMS bindings. It ensures delivery of one-way messages by wrapping message operations (send or receive) within transactions. JMS Transacted OneWay policy is supported only when the configured JMS provider is an Enterprise Message Service.

The JMS service binding starts a transaction by enlisting an incoming JMS message. The transaction context is passed to subsequent components, ensuring that the operations from these components are part of the transaction. The operations include: database operations performed by components such as Java, Spring, Mediation, and BWSE and the outgoing JMS message from the JMS reference binding.

When control returns to the JMS service binding, the transaction is either committed, if the provider returns normally, or it is rolled back, if the provider throws an undeclared fault or exception.

At a JMS service binding, this policy set ensures that the binding receives each inbound one-way message from the JMS destination within a transaction started by the JMS service binding. If the transaction rolls back, the message remains in the original JMS destination, which can be redelivered to the JMS service binding within a subsequent transaction. If redelivery succeeds and control is returned to the JMS binding successfully, the JMS message on the source destination will be committed.

If re-delivery attempts are exhausted, then the message will be moved to the configured error-queue (if it is configured) in the same transaction. If the error-queue is not configured, the message is discarded.


At a JMS reference binding, this policy set ensures that the JMS binding sends each outbound one-way message within the component's open transaction.

This policy set can apply to any JMS binding on a promoted service or reference.

- The JMS binding and the components wired to it must be deployed to the single node to achieve a transaction that includes the incoming JMS message, the database operations, and the outgoing JMS message from the JMS binding on the reference side.
- The components participating in the transaction must be configured with a Managed Global Transaction policy.
- If an error queue has not been created and the JMS Provider does not support dynamic queue creation or the user of the provider does not have create permissions, JMS binding deployment fails. If the error queue is deleted after the application is deployed, JMS binding attempts to move the erroneous message to the error queue. On failing to do so, it indefinitely attempts to move the message to the error queue regardless of the redelivery attempt configuration.



Parameter	Description
Max redelivery count	<p>The JMS binding continues to attempt delivery until either delivery succeeds, or the number of delivery attempts reaches this value. This value specifies the total number of times the message is delivered, including the first delivery.</p> <p>For example:</p> <ul style="list-style-type: none"> • If the value is 1, the Messaging Bus attempts only the initial delivery and then stops. • If the value is 3, the sequence is as follows: <ol style="list-style-type: none"> 1. The message is delivered once. (first attempt) 2. On failure, the message is delivered again. (second attempt) 3. On failure, the message is delivered again. (third attempt) 4. A failure is reported. <p>Default: 0.</p>
Redelivery delay interval (s)	<p>After each failed delivery attempt, the JMS binding waits for this interval before its next attempt.</p> <p>Default: 15.</p>

Parameter	Description
Exceptions upon which to stop redelivery	<p>If the provider throws any exception in this list, then the JMS binding cancels further delivery attempts (even if the maximum redelivery count has not yet reached).</p> <p>Default: None.</p>
Enable error queue	<p>Indicate how a message is treated when delivery fails—that is, the JMS binding exceeds the maximum delivery count or the service implementation throws an error that stops delivery.</p> <p>When checked, place the message on the error queue defined in the Error queue name field. When unchecked, discard the message.</p> <p>Default: Unchecked.</p>
Transactional	<p>Indicates whether move to error queue should happen in the same session transaction in which message from source destination was received.</p> <div>  <p>Transaction here is not an XA transaction, this is only a session transaction.</p> </div>
Error queue name	<p>The queue on which JMS binding places failed message if the error queue is enabled.</p> <p>A warning is displayed on the Promoted Service indicating configured error queue should be present on the JMS provider.</p> <p>Default: Default error queue name is JMSAtLeastOnce_ErrorQueue.</p>

Managed Transaction

A Managed Transaction policy starts a managed transaction before invoking a component service, so its service implementation executes in a transaction environment.

TIBCO ActiveMatrix starts or joins (inherits) a transaction before invoking the operation of a component service. When control returns to TIBCO ActiveMatrix, the transaction either commits (for example, if the component returns normally or throws a business exception (faults declared in the abstract WSDL)) or rolls back (for example, if the component throws a non-business exception).

TIBCO Business Studio automatically adds this policy set when it adds the corresponding intent.

This policy set can apply at any implementation.

Message Transmission Optimization Mechanism (MTOM)

A MTOM policy enables you to configure how binary data (attachments such as images, PDF files, and so on) is to be transmitted over SOAP.

When the MTOM policy is applied, the SOAP runtime optimizes the binary data transmission using XML-binary Optimized Packaging (XOP).

- **Outbound messages:** The binary data for outbound messages (only if the payload has base64binary elements) is optimized.
- **Inbound messages:** For incoming messages, regardless of the policy, SOAP runtime infers whether the payload is optimized or not and acts based on that information. That is, when a SOAP service receives a request message or SOAP reference receives a reply message (or business fault) and if that message is optimized, the optimized binary MIME data is converted into Base64 encoded data regardless of the MTOM policy.

This policy can be applied to SOAP services and reference bindings for:

- SOAP over HTTP
- TIBCO SOAP over JMS
- W3C SOAP over JMS

For additional information, refer to:

- <http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/>
- <http://www.w3.org/TR/2005/REC-xop10-20050125/>

Prepare Before Undeploy

A Prepare Before Undeploy policy ensures that Administrator does not undeploy an application or component until the application or component explicitly confirms that its preparations are complete.

This policy set can apply at any implementation.

Secure Virtualization

Ensures that a service receives all messages from the Messaging Bus using SSL.

This policy set can apply at a Virtualization binding on a service.

Start Service First

Ensures that when a reference is wired to a service, the provider component starts before the referencing component starts.

This policy set can apply at any reference that is wired to a service.

Threading

Ensures that message dispatch and send operations use a thread pool.

A service uses the thread pool to dispatch each inbound message.

A reference uses the thread pool to send each outbound message.

This policy set can apply at a Virtualization binding on a service or reference, or at any SOAP binding on a promoted service or reference.

Threading policy at a service or reference automatically gets propagated to the promoted service or reference.

Parameter	Description
Thread Pool Resource Template	A Thread Pool resource template. A thread pool resource instance must be installed on each node on which a component or binding with a threading policy is distributed. Default: None.

Parameter	Description
Invocation Timeout	<p>Send operations at references must complete before the invocation timeout, otherwise the operation throws the exception <code>com.tibco.amf.platform.runtime.extension.exception.TimeoutException</code>.</p> <ul style="list-style-type: none"> For messages that require a response, the reference must receive the response before the timeout. For one-way messages, the reference must complete the send operation before the timeout. <p>Zero is a special value, indicating no timeout on the send operation.</p> <p>Default: 0 s.</p>


Virtualization At Least Once

Ensures that the Messaging Bus attempts to deliver each message to its destination at least once.

Delivery succeeds when the provider acknowledges receipt. Otherwise, if a delivery attempt fails, the Messaging Bus attempts to redeliver according to the parameters of this policy set. If the parameters preclude redelivery, the entire message fails, and other parameters determine message disposal.

The Messaging Bus preserves messages in its persistent store until success or disposal. The Messaging Bus does not deliver a message within a transaction (however, the provider may independently process and acknowledge the message with a transaction).

This policy set can apply at a Virtualization binding on a service or reference, or at a SOAP binding on a promoted service or reference.

Parameter	Description
Max redelivery count	<p>The Messaging Bus continues to attempt delivery until either delivery succeeds, or the number of delivery attempts reaches this value. For example, with a value of 1, the Messaging Bus attempts only the initial delivery, then stops.</p> <p>Zero is a special value, indicating no upper limit on redelivery.</p> <p>Default: 0</p>
Redelivery delay interval (s)	<p>After each failed delivery attempt, the Messaging Bus waits for this interval before its next attempt.</p> <div>  <p>The redelivery time is not implemented exactly. The difference between the specified delay and the actual delay can be up to an additional 15 s.</p> </div> <p>Default: 15.</p>
Exceptions upon which to stop redelivery	<p>If the provider throws any exception in this list, then the Messaging Bus cancels further delivery attempts (even if the maximum is not yet reached).</p> <p>Default: None.</p>

Parameter	Description
Enable error queue	<p>This parameter determines the disposal of a message when delivery fails—that is, the Messaging Bus exceeds the maximum delivery count or the provider throws an error that stops delivery.</p> <p>When enabled, place the message on the error queue (see below).</p> <p>When disabled, discard the message.</p> <p>Default: Disabled.</p>
Error queue name	<p>The Messaging Bus places failed messages on this queue (if the error queue is enabled).</p> <p>Default: Create the default error queue by appending <code>_ErrorQueue</code> to the service queue name.</p>
Transactional	<p>When the error queue is enabled, the Messaging Bus does two operations after each failed message:</p> <ol style="list-style-type: none"> 1. Place the failed message on the error queue. 2. Remove the failed message from the provider queue. <p>When this parameter is disabled, the error queue can contain duplicate copies of the failed message. For example, if a hardware failure interrupts between the two operations, then after restart the message could place a second copy on the error queue.</p> <p>When this parameter is enabled, the Messaging Bus prevents duplicates on the error queue by wrapping the two operations in a transaction. If a hardware failure interrupts between the two operations, the transaction rolls back after restart.</p> <p>Transactions dramatically decrease message delivery performance. Unless it is very important to prevent duplicate failed messages on the error queue, we recommend disabling this parameter.</p> <p>Default: Disabled.</p>

Virtualization Transacted OneWay

Ensures that delivery of one-way messages are included as part of a transaction.

TIBCO ActiveMatrix starts a transaction, which encompasses all message processing operations—such as receiving a message and invoking operations at other providers—and also operations such as database updates (if needed). When control returns to TIBCO ActiveMatrix, the transaction either commits (for example, if the provider returns normally) or rolls back (for example, if the provider throws an exception).


At a service, this policy set ensures that the Messaging Bus receives each inbound one-way message from the transport binding within the component's open transaction. If the transaction rolls back, the message remains in the transport queue, which can redeliver the message within a subsequent transaction.

At a reference, this policy set ensures that the transport sends each outbound one-way message within the component's open transaction.

From the perspective of the service queue, if the transaction commits, delivery succeeds, and the Messaging Bus removes the message from the queue. Otherwise, if the transaction rolls back, the delivery attempt fails, and the Messaging Bus attempts to redeliver according to the parameters of this

policy set. If the parameters preclude redelivery, the entire message fails, and other parameters determine message disposal.

This policy set can apply at a Virtualization binding on a service or reference, or at a SOAP binding on a promoted service or reference. However, if this policy is applied to a SOAP binding it will take effect on the virtualization proxy between the binding and the component.

Parameter	Description
Max redelivery count	<p>The Messaging Bus continues to attempt delivery until either delivery succeeds, or the number of delivery attempts reaches this value. For example, with a value of 1, the Messaging Bus attempts only the initial delivery, then stops.</p> <p>Zero is a special value, indicating no upper limit on redelivery.</p> <p>Default: 0.</p>
Redelivery delay interval (s)	<p>After each failed delivery attempt, the Messaging Bus waits for this interval before its next attempt.</p> <div>  <p>The redelivery time is not implemented exactly. The difference between the specified delay and the actual delay can be up to an additional 15 s.</p> </div> <p>Default: 15 s.</p>
Exceptions upon which to stop redelivery	<p>If the provider throws any exception in this list, then the Messaging Bus cancels further delivery attempts (even if the maximum is not yet reached).</p> <p>Default: None.</p>
Enable error queue	<p>This parameter determines the disposal of a message when delivery fails—that is, the Messaging Bus exceeds the maximum delivery count or the service implementation throws an error that stops delivery.</p> <p>When enabled, place the message on the error queue (see below).</p> <p>When disabled, discard the message.</p> <p>Default: Disabled.</p>
Error queue name	<p>The Messaging Bus places failed messages on this queue (if the error queue is enabled).</p> <p>Default: Create the default error queue by appending <code>_ErrorQueue</code> to the service queue name.</p>

Virtualize

Ensures that messages travel through the Messaging Bus, blocking potential optimization, while enabling beneficial side effects of the Messaging Bus.

When a consumer and provider run in the same node, the component framework ordinarily attempts to optimize message delivery. Bypassing the Messaging Bus, it transmits the message by passing a memory pointer. This policy set blocks that optimization, and guarantees that messages travel through the Messaging Bus.

The Messaging Bus lets several components receive messages from the same destination, which results in a simple load-balancing effect. Optimization bypasses the Messaging Bus, which precludes such load-balancing. Use this policy to allow load-balancing.

This policy set can apply at a Virtualization binding on a service.

Never Virtualize

You can add this policy in TIBCO Business Studio to a composite, on a service/reference binding, component service or a component reference.

This policy enforces that messages between components (or between a binding and a component) are always routed via in-memory transport, and not via the virtualization queues that use the Enterprise Message Server (EMS). If the target component is not available and hence the message cannot be routed in-memory, the invocation fails with an exception. When used in conjunction with a transaction policy, this will cause the transaction to be rolled back.

This policy supports a specific use-case where a SOAP/JMS or JMS binding with a 'JMS Transacted OneWay' policy wired to several components and managed by a single transaction. In this use-case, a transaction starts at the SOAP/JMS binding and spans several components, and it is expected that if the transaction rolls back, the incoming messages are returned to the external JMS queue.

As an example, suppose we have a component C1 wired to C2. When C2 is stopped, it will result in an exception, causing the invocation from C1 to C2 to fail. Without the 'Never Virtualize' policy, the message would have been routed to an internal virtualization queue for future delivery when C2 is back up and running.

The 'Never Virtualize' policy has the opposite behavior of the 'Virtualize' policy. As such, both these policies can be used with or without a transaction policy, and both control the internal message routing.

Comparison between these policies as applied to the example of C1 wired to C2:

- Never Virtualize policy
 - Forces the use of in-memory transport.
 - C1 and C2 must be co-located in the same node for this policy to be useful.
 - Causes C1's invocation to fail when C2 is down/absent on the same node.
- Virtualize policy
 - Forces the use of EMS-based transport using an internal virtualization queue.
 - Invocations work when C1 and C2 are on the same or different nodes.
 - Causes C1's invocation to be blocked when C2 is down/unavailable. The message is delivered later when C2 comes back up.
 - Cannot be used for the above use-case since virtualized queues are always used.
- Default (neither of the two policies declared):
 - Automatically chooses whether to use in-memory routing or EMS-based routing depending on whether C2 is running or not on the same node as C1.
 - Cannot be used for the above use-case since virtualized queues will be used when C2 is unavailable

When the 'Never Virtualize' policy is used together with a transaction policy, ensure that the transaction policy be configured with a finite number of attempts for message delivery. For example, while using 'JMS Transacted OneWay' policy, the 'Max redelivery count' parameter must have a finite non-zero value. Using the default value of zero implies infinite re-delivery attempts and will cause the system redelivering messages to itself continuously and infinitely when one of the components is stopped or the transaction rolls back due to any other reason.

WS-Addressing for References

A WS-Addressing for Reference policy set ensures that a reference sends messages with WS-Addressing semantics.

You must configure WS-Addressing policies in pairs at references and services. The values of the Anonymous parameters of the two policies must match.

This policy set can apply at a SOAP binding on a promoted reference.

Element	Description
Anonymous	<p>When anonymous is enabled, the reference expects synchronous responses to its requests. The reference waits for an immediate reply on the same HTTP channel as the request.</p> <p>When anonymous is disabled, the reference expects asynchronous responses to its requests. The reference waits for an immediate acknowledgment on the same HTTP channel as the request, and is ready to receive an asynchronous response at the addresses specified in the <code>replyTo</code> or <code>faultTo</code> header elements of the request message.</p> <p>Default: Anonymous.</p>

WS-Addressing for Services

A WS-Addressing for Services policy ensures that a service receives messages with WS-Addressing semantics.

You must configure WS-Addressing policies in pairs at references and services. The values of the Anonymous parameters of the two policies must match.

This policy set can apply at a SOAP binding on a promoted service.

Element	Description
Anonymous	<p>When anonymous is enabled, the service sends synchronous responses to requests. The service sends an immediate reply on the same HTTP channel as the request.</p> <p>When anonymous is disabled, the service sends asynchronous responses to requests. The service sends an immediate acknowledgment on the same HTTP channel as the request, and later sends an asynchronous response to the addresses specified in the <code>replyTo</code> or <code>faultTo</code> header elements of the request message.</p> <p>Default: Anonymous.</p>

WS-Reliable Messaging

A WS-Reliable Messaging policy ensures that a service receives messages or a reference sends messages with WS-Reliable Messaging semantics.

This policy set can apply at a SOAP binding on a promoted service or reference.

Element	Description
Reliability	Select from the available WS-Reliable Messaging delivery semantics. Default: Exactly Once.

Policy Templates Reference

You can configure these policies by copying a template into an external policy set, and modifying the parameters appropriately.

You can find sample templates in the file *TIBCO_HOME/ amx/version/ samples/policy/ samples.zip*.

Category	Template Description
Authorization	Authorization By Role Policies
Authentication	Basic Authentication Policies
	Basic Or Username Token Authentication Policies
	SAML Authentication For SSO Policies
	Username Token Authentication Policies
Credential Mapping	Basic Credential Mapping Policies
	SAML Credential Mapping For SSO Policies
WS-Security	WS-Security Consumer Policies
	WS-Security Provider Policies

Policy Template to Intents Reference

The intents that a policy can provide is a subset of the intents that the policy template can provide; the policy configuration can narrow that set.

The intents that each policy template can provide are listed below.

Policy Set Template	Can Provide these Intents
Authorization By Role Policies	scaext:authorization.role
Basic Authentication Policies	scaext:clientAuthentication.basic
Basic Credential Mapping Policies	scaext:credentialMapping.basic
Basic Or Username Token Authentication Policies	scaext:clientAuthentication.basic scaext:clientAuthentication.usernameToken
SAML Authentication For SSO Policies	scaext:clientAuthentication.ssoSAML

Policy Set Template	Can Provide these Intents
SAML Credential Mapping For SSO Policies	scaext:credentialMapping.ssoSAML
Username Token Authentication Policies	scaext:clientAuthentication.usernameToken
WS-Security Consumer Policies	scaext:credentialMapping.wssSAML scaext:credentialMapping.usernameToken scaext:consumerIntegrity.wss scaext:consumerConfidentiality.wss
WS-Security Provider Policies	scaext:clientAuthentication.wssSAML scaext:clientAuthentication.usernameToken scaext:clientAuthentication.x509 scaext:providerIntegrity.wss scaext:providerConfidentiality.wss

TIBCO Business Studio lets you specify several security intents on a binding or component. For simplicity, we recommend satisfying those intents with fewer policies and policy sets (rather than proliferating many). That is, where possible, use policies that satisfy several intents.

The policy samples in *TIBCO_HOME/amx/version/samples/policy/samples.zip* represent some typical use cases. They are organized in subdirectories by policy template name.

Authorization By Role Policies

You can configure Authorization By Role policies by copying a template into an external policy set, and modifying the parameters appropriately. You can find sample templates in an archive file under *TIBCO_HOME/amx/version/samples/policy/samples.zip*

Several template samples are available.

Template File
AllOperationsAllowedForRole.policysets
AuthenticatedUsersOnly.policysets
EveryoneAllowed.policysets
NobodyAllowed.policysets
SpecificOperationAllowedForALLRoles.policysets
SpecificOperationSpecificRole.policysets
Can Provide these Intents
scaext:authorization.role

Basic Authentication Policies

You can configure the Basic Authentication policy by copying a template into an external policy set, and modifying the parameters appropriately. You can find a sample template in an archive file under *TIBCO_HOME/amx/version/samples/policy/samples.zip*

Template File
BasicAuthenticationWithWebAppUsingLDAP.policysets
Can Provide these Intents
scaext:clientAuthentication.basic

Basic Credential Mapping Policies

You can configure Basic Credential Mapping policies by copying a template into an external policy set, and modifying the parameters appropriately. You can find sample templates in an archive file under *TIBCO_HOME/amx/version/samples/policy/samples.zip*

You can configure this policy to retrieve user credentials from an Identity Provider resource instance. When using an Identity Provider resource instance to retrieve user credentials for a policy, in the Identity Provider resource template, check the Enable Access to Credential Store Containing Identity checkbox. The JCEKS keystore used in the Identity Provider resource template should be able to store symmetric keys.

Several template samples are available.

Template File
BasicCredentialMappingFixed.policysets
BasicCredentialMappingRoleBased.policysets
Can Provide these Intents
scaext:credentialMapping.basic

UsernameToken - Nonce and Created Elements

When a Basic Credential Mapping or WSS Credential Mapping policy is used to insert a UsernameToken in the SOAP security header, the Nonce and Created elements can be optionally added.

You can configure a Basic Credential Mapping or WS-Security Consumer Credential Mapping policy to have the UsernameToken without the Nonce and Created elements by copying the template below and modifying the parameters appropriately. See the Policy Sets, Policy Templates Reference section in the Composite Development guide for more information about configuring policy sets.

The sample Basic Credential Mapping policy below generates the UsernameToken without the Nonce and Created elements.

```
<?xml version="1.0" encoding="UTF-8"?>
<ep:policySetContainer xmlns:ep="http://xsd.tns.tibco.com/amf/models/
externalpolicy"
```

```

xmlns:sca="http://www.oso.org/xmlns/sca/1.0"
xmlns:scaext="http://xsd.tns.tibco.com/amf/models/sca/extensions"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wssp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
seceext-1.0
.xsd"
xmlns:tpa="http://xsd.tns.tibco.com/governance/policy/action/2009"
xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
xmlns:tpc="http://xsd.tns.tibco.com/governance/policy/common/2009"
xmlns:jmsbt="http://xsd.tns.tibco.com/amf/models/sca/bindingtype/jms"
xmlns:soapbt="http://xsd.tns.tibco.com/amf/models/sca/bindingtype/soap"
xmlns:webapp="http://xsd.tns.tibco.com/amf/models/sca/implementationtype/webapp"
targetNamespace="http://www.example.org">

<!-- add the policy sets here -->
<sca:policySet name="CredentialMappingUsernameToken"
provides="scaext:clientAuthentication.usernameToken"
appliesTo="soapbt:binding.soap.service">
  <wsp:Policy template="tpt:WssConsumer" xmlns:tpt="
http://xsd.tns.tibco.com/governance/policy/template/2009">
    <wsp:All>
      <wsp:Policy>
        <wsp:All>
          <tpa:CredentialMapping>
            <tpa:Fixed>
              <wssp:UsernameToken>
                <wsse:Username>schalla</wsse:Username>
                <wsse:Password>password</wsse:Password>
              </wssp:UsernameToken>
              <tpa:IdentityProvider
ResourceInstance="IdPasswordProvider" />
            </tpa:Fixed>
            <wssp:SupportingTokens>
              <wssp:UsernameToken>
                <tpa:NoNonce/>
              </wssp:UsernameToken>
            </wssp:SupportingTokens>
          </tpa:CredentialMapping>
        </wsp:All>
      </wsp:Policy>
    </wsp:All>
  </wsp:Policy>
</sca:policySet>
</ep:policySetContainer>

```

Basic Or Username Token Authentication Policies

You can configure the Basic Or Username Token Authentication policy by copying a template into an external policy set, and modifying the parameters. You can find a sample template in an archive file under *TIBCO_HOME/amx/version/samples/policy/samples.zip*

One template sample is available.

Template File
BasicOrUsernameTokenAuthenticationWithSoapEpUsingLDAP.policysets
Can Provide these Intents
scaext:clientAuthentication.basic
scaext:clientAuthentication.usernameToken

SAML Authentication For SSO Policies

You can configure SAML Authentication For SSO Policies by copying a template into an external policy set, and modifying the parameters. You can find sample templates in an archive file under *TIBCO_HOME/amx/version/samples/policy/samples.zip*.

Component services or promoted references authenticate the consumer's identity using a single sign-on SAML token. (Credential mapping policies propagate the SAML token to providers within the ActiveMatrix environment.

Several template samples are available.

Template File
SAMLAAuthenticationForSSOSigned.policysets
SAMLAAuthenticationForSSOUnsigned.policysets
Can Provide these Intents
scaext:clientAuthentication.ssoSAML

SAML Credential Mapping For SSO Policies

You can configure SAML Credential Mapping For SSO policies by copying a template into an external policy set, and modifying the parameters. You can find sample templates in an archive file under *TIBCO_HOME/amx/version/samples/policy/samples.zip*.

References (or promoted services) propagate a SAML token asserting the consumer's identity to providers within the ActiveMatrix environment.

Several template samples are available.

Template File
SAMLCredentialMappingForSSOSigned.policysets
SAMLCredentialMappingForSSOUnsigned.policysets
Can Provide these Intents
scaext:credentialMapping.ssoSAML

Username Token Authentication Policies

You can configure Username Token Authentication policies by copying a template into an external policy set, and modifying the parameters. You can find a sample template in an archive file under *TIBCO_HOME/amx/version/samples/policy/samples.zip*.

One template sample is available.

Template File
UsernameTokenAuthenticationWithSoapEpUsingLDAP.policysets

Can Provide these Intents
scaext:clientAuthentication.usernameToken

WS-Security Consumer Policies

You can configure WS-Security Consumer policies by copying a template into an external policy set, and modifying the parameters. You can find sample templates in an archive file under *TIBCO_HOME/amx/version/samples/policy/samples.zip*.

Several template samples are available.

You can configure this policy to retrieve user credentials from an Identity Provider resource instance. When using an Identity Provider resource instance to retrieve user credentials for a policy, in the Identity Provider resource template, check the Enable Access to Credential Store Containing Identity checkbox. The JCEKS keystore used in the Identity Provider resource template should be able to store symmetric keys.

Template File
WssConsumerAddUsernameTokenTimestampSignAndEncrypt.policysets
WssConsumerCredentialMappingSAMLSigned.policysets
WssConsumerCredentialMappingSAMLUnsigned.policysets
WssConsumerCredentialMappingUsernameTokenFixed.policysets
WssConsumerCredentialMappingUsernameTokenRoleBased.policysets

Can Provide these Intents
scaext:credentialMapping.wssSAML
scaext:credentialMapping.usernameToken
scaext:consumerIntegrity.wss
scaext:consumerConfidentiality.wss

UsernameToken - Nonce and Created Elements

When a Basic Credential Mapping or WSS Credential Mapping policy is used to insert a UsernameToken in the SOAP security header, the Nonce and Created elements can be optionally added.

You can configure a Basic Credential Mapping or WS-Security Consumer Credential Mapping policy to have the UsernameToken without the Nonce and Created elements by copying the template below and modifying the parameters appropriately. See the Policy Sets, Policy Templates Reference section in the Composite Development guide for more information about configuring policy sets.

The sample Basic Credential Mapping policy below generates the UsernameToken without the Nonce and Created elements.

```
<?xml version="1.0" encoding="UTF-8"?>
<ep:policySetContainer xmlns:ep="http://xsd.tns.tibco.com/amf/models/
```

```

externalpolicy"
  xmlns:sca="http://www.oso.org/xmlns/sca/1.0"
  xmlns:scaext="http://xsd.tns.tibco.com/amf/models/sca/extensions"
  xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
  xmlns:wssp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
  xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secect-1.0
.xsd"
  xmlns:tpa="http://xsd.tns.tibco.com/governance/policy/action/2009"
  xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
  xmlns:tpc="http://xsd.tns.tibco.com/governance/policy/common/2009"
  xmlns:jmsbt="http://xsd.tns.tibco.com/amf/models/sca/bindingtype/jms"
  xmlns:soapbt="http://xsd.tns.tibco.com/amf/models/sca/binding/soap"
  xmlns:webapp="http://xsd.tns.tibco.com/amf/models/sca/implementationtype/webapp"
  targetNamespace="http://www.example.org">

<!-- add the policy sets here -->
<sca:policySet name="CredentialMappingUsernameToken"
provides="scaext:clientAuthentication.usernameToken"
appliesTo="soapbt:binding.soap.service">
  <wsp:Policy template="tpt:WssConsumer" xmlns:tpt="
http://xsd.tns.tibco.com/governance/policy/template/2009">
    <wsp:All>
      <wsp:Policy>
        <wsp:All>
          <tpa:CredentialMapping>
            <tpa:Fixed>
              <wssp:UsernameToken>
                <wsse:Username>schalla</wsse:Username>
                <wsse:Password>password</wsse:Password>
              </wssp:UsernameToken>
              <tpa:IdentityProvider
                ResourceInstance="IdPasswordProvider" />
            </tpa:Fixed>
            <wssp:SupportingTokens>
              <wssp:UsernameToken>
                <tpa:NoNonce/>
              </wssp:UsernameToken>
            </wssp:SupportingTokens>
          </tpa:CredentialMapping>
        </wsp:All>
      </wsp:Policy>
    </wsp:All>
  </wsp:Policy>
</sca:policySet>
</ep:policySetContainer>

```

WS-Security Provider Policies

You can configure WS-Security Provider policies by copying a template into an external policy set, and modifying the parameters. You can find sample templates in an archive file under *TIBCO_HOME/amx/version/samples/policy/samples.zip*.

Template File
WssProviderAuthenticateSAMLSigned.policysets
WssProviderAuthenticateSAMLUnsigned.policysets
WssProviderAuthenticateUsernameTokenAndTimestamp.policysets
WssProviderDecryptAuthenticateUsernameTokenAndSignatureTimestamp.policysets

Can Provide these Intents
scaext:clientAuthentication.wssSAML
scaext:clientAuthentication.usernameToken
scaext:clientAuthentication.x509
scaext:providerIntegrity.wss
scaext:providerConfidentiality.wss

Transactions

TIBCO ActiveMatrix support for transactions conforms to the OASIS Service Component Architecture Policy specification and supports several transaction types.

The following types of transactions are supported:

- Managed Global
- OneWay

The database operations performed in a component implementation can participate in a transaction only if the following conditions must be satisfied:

- The component implementation must use a database connection provided by a JDBC resource instance.
- The connection type of the JDBC resource template must be XA.

Managed Global Transactions

A managed global transaction provides transaction support for database operations performed in one or more Spring, Java, Mediation, and BWSE component implementations. Database operations performed under managed global transaction in multiple components are atomically committed or rolled back. A managed global transaction conforms to the SCA managedTransaction.global intent.

A managed global transaction is started and terminated by the TIBCO ActiveMatrix platform. Managed global transactions are enabled by configuring a Managed Global Transaction intent and policy on each component that needs the transaction.

The TIBCO ActiveMatrix platform starts a new transaction for a component if:

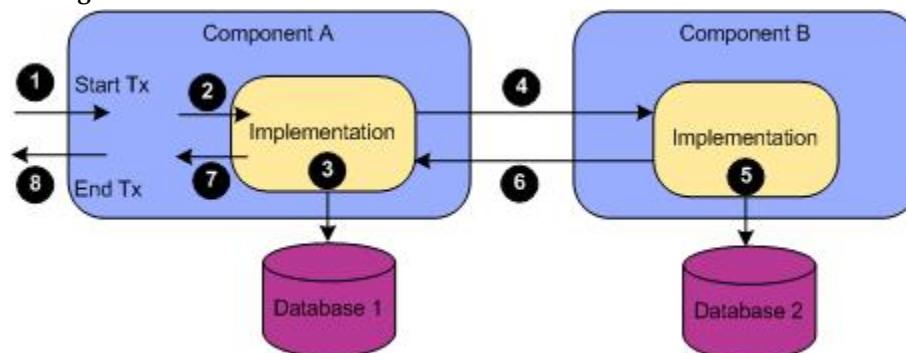
- A service of the component is invoked.
- The component is configured with a Managed Global Transaction policy.
- The component service is configured with a Transacted OneWay policy.
- The thread used to invoke the component does not have a transaction already in progress.

The TIBCO ActiveMatrix platform propagates the transaction through the component references unless:

- The component reference is configured with Transacted OneWay, AtLeastOnce, or Virtualize policy.
- The target of the component reference is running on a different node.

The following figure illustrates the flow between the participants in a managed global transaction.

Managed Global Transaction Flow



In this scenario, the transaction participants are:

- The operations on Database 1 (3).
- The operations on Database 2 (5).

If a managed global transaction is in progress and the conditions stated above are satisfied, then the transaction will propagate to the called component even if the called component is not enabled with a Managed Global Transaction intent.

Runtime Behavior

When there is no propagated transaction:

- A new managed global transaction is started by TIBCO ActiveMatrix before the request message is delivered to service implementation.
- For In-Out MEPs, the transaction is terminated by TIBCO ActiveMatrix on return from invocation of the service implementation. For In-Only MEPs, the transaction is terminated when the request processing completed by the service implementation.
 - The transaction is committed when the service returns without exceptions or throws a business exception (that is, a fault declared in the WSDL file).
 - The transaction is rolled back when the service throws an unchecked or non-declared exceptions.

When a propagated transaction is present:

- The invoked component will participate in the propagated transaction and makes it a global transaction before the request message is delivered to component implementation.
- The transaction is not terminated by the TIBCO ActiveMatrix platform on return from invocation of the component implementation. The transaction is always terminated by the TIBCO ActiveMatrix platform for the component that started the transaction.

Transacted OneWay Transactions

A Transacted OneWay transaction provides transaction support for component service messages, database operations, and component reference messages. OneWay transactions are supported only for In-Only MEPs.

An In-Only MEP message received by a component service, the database operations performed in the component implementation, and the In-Only message sent by component references can be part of a transaction and will be atomically committed or rolled back. The primary application of Transacted OneWay transactions is ensuring that one-way messages are not lost in the course of processing database updates. A Transacted OneWay transaction conforms to the SCA transactedOneWay intent.

Transacted OneWay transactions are enabled by configuring a Transacted OneWay intent and policy on a composite service, component service, component reference, or composite reference that are not simultaneously enabled with any of following intents:

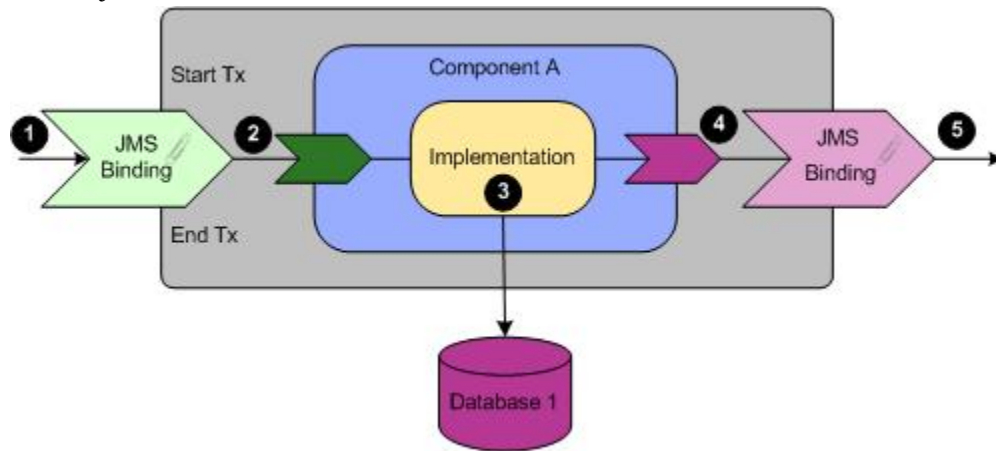
- AtLeastOnce
- AtMostOnce

If the Transacted OneWay intent and policy is configured on a service, a transaction is started and terminated by the TIBCO ActiveMatrix platform. If the component uses non-managed transactions, its implementation starts the transaction, and component has a reference with a Transacted OneWay intent and policy, the reference participates in the transaction.

Scenarios

The following figure illustrates the participants in a OneWay transaction.

OneWay Transaction



The transaction participants are:

- The JMS binding's incoming message (1).
- The component service's incoming message (2).
- The operations on Database 1 (3).
- The component reference's outgoing message (4).
- The JMS binding's outgoing message (5). The message is sent to the referenced service when the transaction is committed.

Runtime Behavior

In-Only messages are exchanged between components via the Messaging Bus even if the components are deployed in the same node.

Component Service Message Processing

If the component service does not have a transaction then:

- The TIBCO ActiveMatrix platform starts a transaction, enlists the request message in the transaction, and delivers the request message to the component implementation.
- If the component implementation successfully processes the request message and does not throw an exception, the transaction is committed.
- If the component implementation throws an exception, the associated Transacted OneWay policy configuration dictates the behavior as follows:
 - The exception is logged.
 - If the exception is one of the exceptions upon which to stop redelivery as configured in the Transacted OneWay policy then:
 1. The transaction is rolled back.
 2. The message is *not* redelivered to the component implementation.
 3. The message is placed in the error queue or discarded based on the corresponding Transacted OneWay Policy configuration.
 4. Processing of the request message is considered complete.
 - If $\text{redeliveryCount} \leq \text{maxRedeliveryCount}$ then the transaction is rolled back and the message is redelivered to the implementation under a new transaction after a delay interval specified in the Transacted OneWay policy.

- If `redeliveryCount > maxRedeliveryCount` then:
 1. The transaction is rolled back.
 2. The message is *not* redelivered to the component implementation.
 3. The message is placed in the error queue or discarded based on the corresponding Transacted OneWay policy configuration.
 4. Processing of the request message is considered complete.

Component Reference Message Processing

If a transaction is active:

- The request message is enqueued as part of the transaction and placed in the target queue.
- If the request message is placed in the target queue successfully then the control is returned to component implementation without any errors; otherwise an exception is returned to the component implementation.

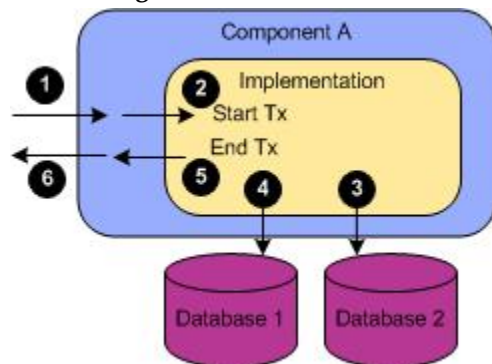
If a transaction is not active, an exception is returned to the component implementation.

Non-Managed Transactions

Non-managed transactions provide transaction support for database operations performed in Spring and Java component implementations. A non-managed transaction must be explicitly started and committed by the component implementation. TIBCO ActiveMatrix guarantees that all database operations are atomically committed or rolled back.

The following figure illustrates the flow between the participants in a non-managed transaction.

Non-managed Transaction Flow



1. In Step 2, the component implementation starts the transaction.
2. Steps 3 and 4 participate in the transaction.
3. In Step 5, the component implementation commits or rolls back the transaction before returning.

Steps 2 and 4 must be performed using the [Java Transaction API](#). To explicitly manage transactions boundaries in a Java component you can retrieve a `javax.transaction.UserTransaction` instance from the component context:

```
import javax.transaction.UserTransaction;
...
UserTransaction tx = componentContext.getUserTransaction();
tx.begin();
...
```

Resource Templates

A *resource template* specifies configuration details for resources. One resource template can be used to configure many resource instances. Resource instances allow sharing of resources such as connection pools. They also eliminate the need to provide such details in services, component implementations, and references. Instead, you specify a property of the type of required resource in the service, component, or reference.

When you configure the application for deployment, you map the property to a resource instance available on the node on which the service, component, or reference is deployed.

Resource templates are typically stored in the `Resource Templates` special folder in an SOA project.

Resource Instances

A *Resource Instance* is a runtime object that represents a resource, such as an HTTP, JDBC, or LDAP connection.

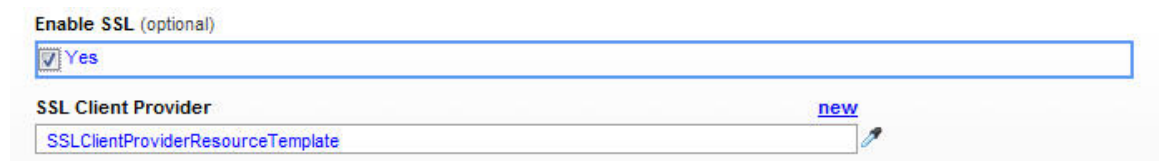
A Resource Instance instantiates the configuration defined in a Resource Template and makes it available to Services running on a Node.

Applications, Components, Bindings, Logging Appenders, and Resource Templates can have properties whose value is the name of a resource. For example, an HTTP client Resource Template's SSL property configuration includes a property whose value is the name of SSL Client Provider resource. [TIBCO Business Studio Resource Instance](#) and [ActiveMatrix Administrator Resource Instance](#) show how to set a resource property in TIBCO Business Studio and ActiveMatrix Administrator respectively.

TIBCO Business Studio Resource Instance



ActiveMatrix Administrator Resource Instance



Creating a Resource Template

One resource template can be used to configure many resource instances. You create a resource template with the resource template wizard.

Procedure

1. Open the create a resource template wizard:
 - Select **File > New > Other > TIBCO Resource Templates > RTType**.
 - Right-click *SOAProjectName* > **Resource Templates**. Select **New > RTType**.
 - Click **Create Resource Template** in a resource template picker.
2. Enter or select the parent folder.
3. In the File Name field, accept the default resource file name or type a new one.
4. Click **OK**.
A resource template file with the specified name is created in the specified folder.

5. [Open the resource template](#) in a resource template editor.
6. Fill out the template properties according to [the property type](#).
7. Save the resource template.

Opening a Resource Template

Procedure

- Choose a context and follow the appropriate procedure.

Context	Procedure
Project Explorer	Double-click a resource template file.
Property Sheet	If the value of a property of type resource template is configured with a resource template, click the field label of a resource template property.

Finding References to a Resource Template

One resource template can be used to configure many resource instances. If you need to know which resource instances have been configured by a resource template, you can do so in the resource template editor.

Procedure

1. Open a resource template in the resource template editor.
2. Under the Name field, click the ***n* reference** link, where *n* is the number of references to the resource template.
A list of objects that reference the resource template display in the Search view.

Renaming a Resource Template

You can rename a resource template in a resource editor. The rename action optionally updates all references to the resource template.

Procedure

1. Open the resource template in a resource editor, choose a control, and follow the appropriate procedure.

Control	Procedure
Name	<ol style="list-style-type: none"> 1. In the Name field, type a new name. 2. Press Enter.
Rename reference	<ol style="list-style-type: none"> 1. Click the Rename reference link. 2. In the New name field, type a new name.

2. Click **Preview**.
A preview pane displays with a list of the effects of the change. If the Update references checkbox is checked, the effects on objects that reference the renamed resource are listed.
3. Uncheck any changes you don't want and click **OK**.

The resource template is renamed. If the Update references checkbox was checked, the selected referencing objects are updated.

Resource Templates Reference

[Resource Templates](#) summarizes the resource templates supported by the TIBCO ActiveMatrix platform. Administrator-specific resource templates are not listed here.

Resource Templates

Type	Description
Data	<ul style="list-style-type: none"> • Hibernate - The Hibernate resource template represents a Hibernate resource. Used by component implementations to access databases, the hibernate is a framework that supports storing Java objects in a relational database. Hibernate solves object-relational impedance mismatch by replacing direct database access with high-level, object-handling functions. • JDBC - The JDBC resource template represents a JDBC connection that is used by component implementations to access databases. • Teneo - A Teneo resource is used by component implementations to access databases. Teneo is a model-relational mapping and runtime database persistence solution for the Eclipse Modeling Framework (EMF). Teneo integrates EMF with Hibernate.
HTTP	<ul style="list-style-type: none"> • HTTP Client - The HTTP Client resource template represents an outgoing HTTP connection. HTTP clients are used by a reference's SOAP binding.
JMS	<ul style="list-style-type: none"> • JMS Resource Templates - JMS resource templates enable applications to access objects maintained in JMS servers.
Security	<ul style="list-style-type: none"> • Security Resource Templates - Security features are provided by a set of resource templates that provide access to various types of security providers: identity, trust, mutual identity, keystore, SSL client and server, and authentication.
External Servers	<ul style="list-style-type: none"> • LDAP Connection - An LDAP Connection resource template represents a connection to an LDAP server. Used by component implementations to look up names in an LDAP directory server. • SMTP - An SMTP resource template represents a connection to an SMTP server. Used by component implementations to send and receive messages to and from an SMTP mail server.
Miscellaneous	<ul style="list-style-type: none"> • Thread Pool - A thread pool is a queue of threads available to run a queue of tasks. Thread pools are used to improve performance when executing large numbers of asynchronous tasks by reducing per-task invocation overhead and provide a means of bounding and managing the resources consumed when executing a collection of tasks.

Resource Templates With Scope

The scope of resource templates can be defined at enterprise level, environment level, and application level.

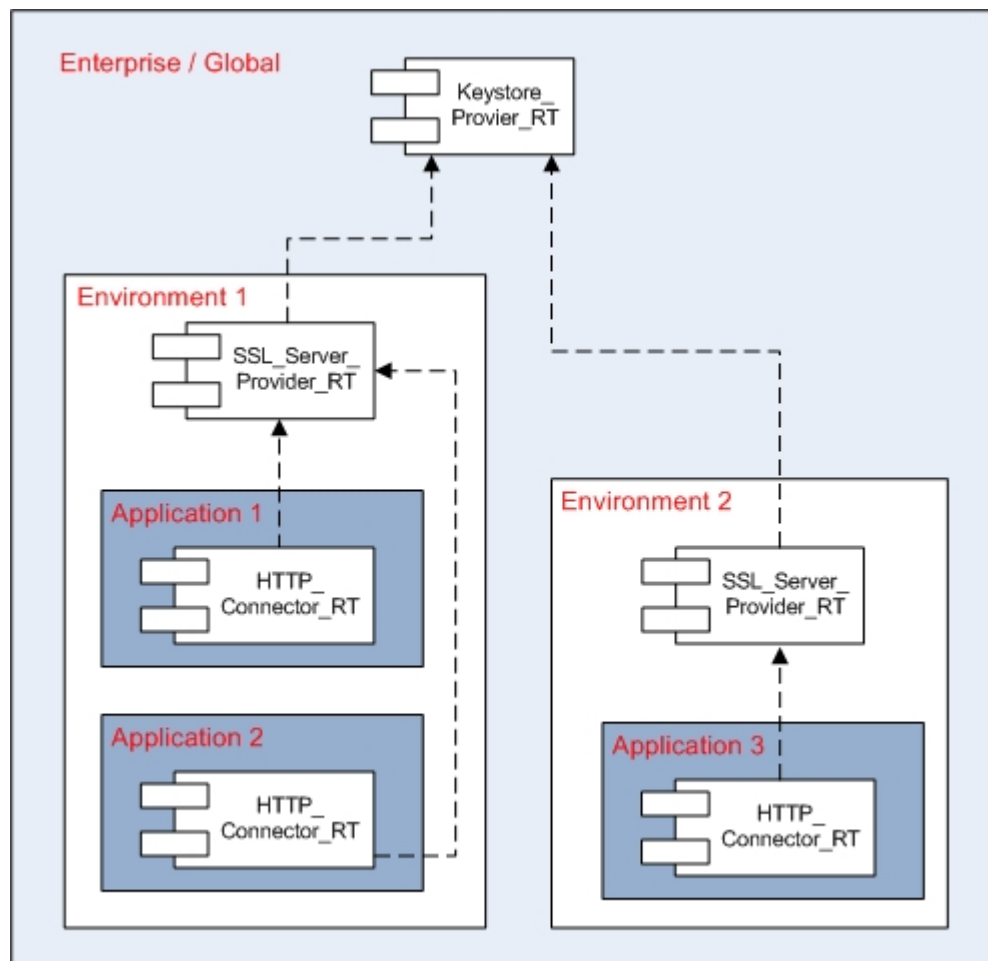
The following levels of scope are available:

- Global or enterprise (default) - available to all environments and applications in the enterprise.
- Environment - available only to applications in a specific environment.
- Application - available only to a specific application running on a node or multiple nodes.

The scope of a resource template is specified at the time of creating it. Later it is possible to change the scope. The following are conditions when changing scope:

- You can specify multiple target elements for a resource template while changing the scope. When multiple target scopes are specified, a resource template in each target scope is created. For example, the resource template with global scope can be scoped to multiple environments or applications.
- If a resource template has a resource instance linked to it, then changing the scope makes a copy than move the resource template itself. For example, if *JDBC_RT* has its scope as global and a *JDBC_RI* linked to it, changing the scope of *JDBC_RT* will make a copy of it for the environment or application than move it to the new scope and remove it from enterprise.

The image below provides an example of how resource templates are scoped at application, environment, and global levels.



Life Cycle of Application Scope Resource Template

When you scope a resource template to an application, the application owns the resource template and the life cycle of the associated resource instances. When you deploy an application:

- Resource instances using the resource templates with scope to the application are installed.
- Resource instances are created in the appropriate nodes as needed.
- A validation process verifies the application property that needs a resource instance matches the resource template name in the application scope. If the match is found the resource instance is automatically created and installed when the application is deployed.

When you undeploy the application all the resource instances using the resource templates with scope defined to the application are uninstalled.

When an application is deleted, all resource templates with scope to the application and the associated resource instances are deleted. This allows creation of an application once and deployment multiple times without conflict.

Uniqueness

- Resource templates names are unique in a specified scope. Two resource templates with the same name cannot exist in the same scope irrespective of the resource template type.
- When a scope is deleted, all resource templates contained in the scope are deleted.
 - When an application is deleted, all resource templates scoped for the application are deleted.
 - Before deleting the resource templates, all resource instances created from the resource templates are un-installed (only relevant for force) and deleted.
- Two applications whose property containing same resource instance name and containing the corresponding resource template configuration cannot coexist on the same node. However, they can both have properties referring to the same JNDI name as long as only one of them provides the application level resource template. For example, consider:
 - Two applications containing a resource template configuration each with same name.
 - When the application is created, the corresponding resource template is created with in its application scope.
 - When deploying the application, this requires two resource instances to be installed with same name, but it cannot because resource instances with same name cannot coexist on the same node.

Resource Dependency and Auto Creation of Resource Instances

- Resource templates can depend on other resources defined in its scope or its parents scope. It cannot depend on its child scope for the purpose of auto creation. However, if dependencies are explicitly created, a resource instance can depend on a resource at a child scope. For example, consider that an HTTP_Client resource template exists in the System Environment which depends on a SSL_Client_Provider in the Enterprise. When HTTP resource instance is created and installed on a node:
 1. It looks for the dependency resource instance (SSL_Client_Provider) on the node
 2. If no resource instance exists, Administrator checks whether an SSL_Client_Provider resource template with the same name exist in the System Environment scope. If the resource template is available at the environment scope, Administrator creates a resource instance using the resource template.

3. If the resource template does not exist in the environment scope, Administrator checks in the Enterprise and creates a resource instance if the resource template is available.

However, if the `SSL_Client_Provider` resource instance with the same name already exists in the specified node, the `HTTP_Client` will depend on it irrespective of its scope.

Permissions

Users need permission to create resource templates and change the scope. For example, a user with permission to an environment can create a resource template to be shared across applications in the environment and not globally at enterprise level.

- Resource templates with global or environment scope have view, edit, and owner permissions that are set individually for each resource templates.
- For global or environment scope, users with Create Resource Template permission can create resource templates at that scope level.
- Resource templates with application scope do not have individual permissions. Users who are granted Manage Resource Template permission for the application can create, edit, view, and delete resource templates with application scope.

Other Conditions

- Resource Templates from one scope are not visible to other scope of the same level. Resource Templates created in `SystemEnvironment` are not visible to `DevEnvironment` and thus are not used for auto creation of resource instances nor do they show up in pickers.
- Resource templates cannot be deleted if a resource instances exists for it.
- Resource instances cannot be uninstalled when other resource instances depend on it.
- Resource instances with the same name cannot coexist in the same node.
- When an application is undeployed, all resource instances using resource templates scoped to that application are uninstalled.
- When an application is deployed, all Resource Templates scoped to that application are:
 1. Checked against the application's properties to see where they are needed.
 2. If specific nodes are identified this way, then Resource Instances with the same name are created and installed on each of those nodes.
 3. If no properties in the application reference the resource templates, then resource instances are automatically created and installed on every node to which the application is mapped.

If a Resource Instance with the required JNDI name already exists on a node where the above rules would cause auto-creation to happen, then deployment validation fails. If redeployment results in the application being removed from a node, all Resource Instances on that node using Resource Templates scoped to the application are uninstalled and deleted.

Changing the Scope of a Resource Template

You can change the scope of a resource template using the Administration GUI. For more information, see [Resource Template With Scope](#).

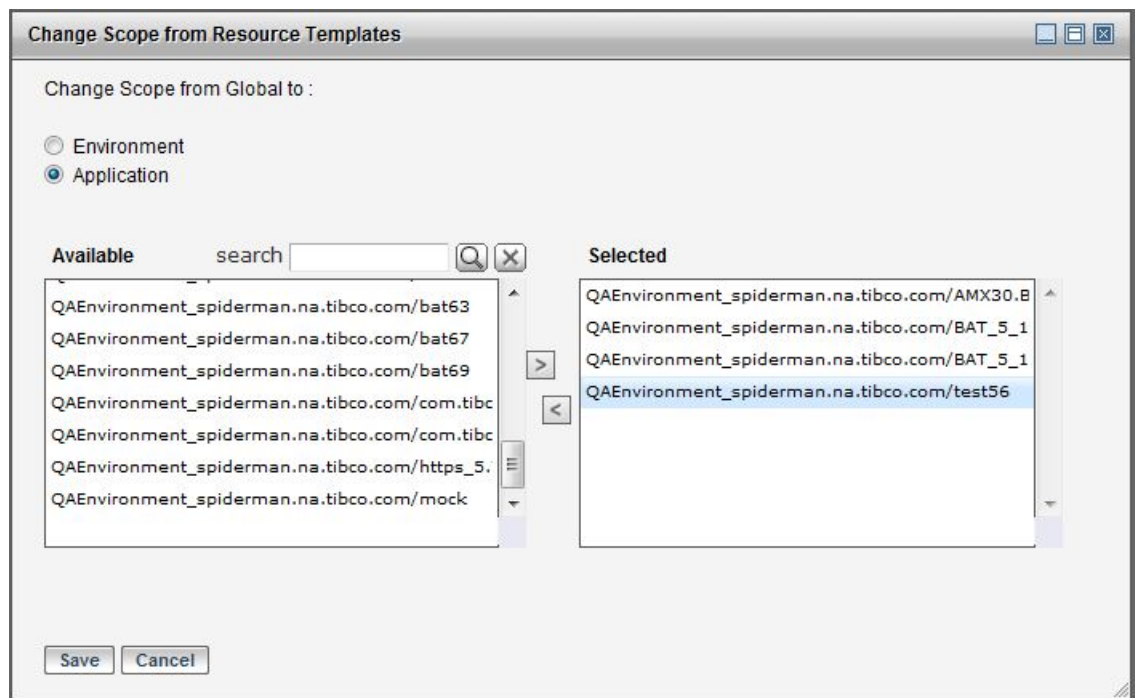
The scope of a resource template can be changed even if:

- a resource instances exists.
- there is a hidden dependency such as inline credentials.
- resource instances on nodes are not related to the target scope.

For more information, see [Resource Template With Scope](#).

Procedure

1. Select **Shared Objects > Resource Templates**.
You can use the Type and Scope to filter the list of resource templates.
2. Select a resource template from the list.
3. Click the **Change Scope**.
Change Scope from Resource Templates window displays.
4. Select Global, Environment, or Application.
 - For a resource template with Global scope, you can change the scope to Environment or Application.
 - For a resource template with Environment scope, you can change the scope to Global or Application.
 - For a resource template with Application scope, you can change the scope to Global or Environment.



Based on the selection, options are displayed in the Available window.

5. Select and use the arrow to move your selection to the Selected window.
6. Click **Save**.

CLI

You can change the scope of a resource template using the command-line utility. You can specify multiple **TargetScope** elements for a resource template.

Procedure

- 1.

- If changing scope from global to environment, specify environment.

```
<TargetScope xsi:type="amxdata_base:Scope" type="Environment"
envName="ENVIRONMENT_NAME"/>
```

- If changing scope from global to application, specify both environment and application.

```
<TargetScope xsi:type="amxdata_base:Scope" type="Application"
envName="ENVIRONMENT_NAME"
appName="APPLICATION_NAME"/>
```

- If changing scope from environment or application to global, specify global.

```
<TargetScope xsi:type="amxdata_base:Scope" type="global"/>
```

The following example shows, the *jdbc_rt* resource template's scope is changed from environment to application *App1*:

```
<Environment xsi:type="amxdata:Environment" name="DevEnvironment" >
  <ResourceTemplate
    xsi:type="amxdata:JdbcResourceTemplate"
    name="jdbc_rt"
    description="Environment jdbc"
    maxConnections="8888">

    <TargetScope xsi:type="amxdata_base:Scope" type="Application"
envName="DevEnvironment" appName="App1"/>

    <Direct
      xsi:type="amxdata:Direct"
      dbUrl="jdbc:hsqldb:sql://localhost:1234/jdbcRtDb"
      jdbcDriver="org.hsqldb.jdbcDriver"
      isTransactional="false"
      loginTimeout="2"/>

    <InlineCredentials username="envJdbc" password="envJdbc"/>

  </ResourceTemplate>
</Environment>
```

2. In the AMXAdminTask element, set the action attribute to changeScope and the objectSelector attribute to ResourceTemplate|Environment/ResourceTemplate|Environment/Application/ResourceTemplate.

```
<AMXAdminTask action="rename"
objectSelector="ResourceTemplate|Environment/ResourceTemplate|Environment/
Application/ResourceTemplate"/>
```

Modifying Resource Templates to Install Multiple BPM Environments

1. Define the scope of the resources using the Administrator.
 - Only one resource instance with a given name can be created on a node and it does not matter from which type of resource template that resource instance was created and what scope that resource template belongs to. For example, let us say you have two resource templates - one Jdbc with scope **Global** and another HttpConnector with scope **Environment**. You can create a resource instance named "ResourceInstance" from Jdbc resource template, on say node "node1". But if you try to create a new resource instance with the same name "ResourceInstance" from HttpConnector resource template, on the same node "node1", this resource instance creation fails.
 - You can have a resource template with identical names and of same type in different environments. And as these resource templates are scoped at the environment, they can have separate configurations.
2. Enforce the configured scope for each shared resource instance.
3. Deploy the application as usual.

Application Deployment Patterns

For an application template, TIBCO ActiveMatrix allows multiple instances of an application to co-exist in an enterprise. Applications use resource instances by declaring a property that can be configured with resource instance name.

The BPM system can be instantiated multiple times on a machine (as a part of the same environment or separate environments), as long as the application instance is distributed to exclusive nodes on the same physical machine. Since the resource instances are configured while deployment of the application instance, they can be instantiated with the same name on disparate nodes.

Multiple resource instances with the same name cannot be instantiated on the same node.

Deploying the application multiple times on nodes belonging to separate environments

With resource templates defined at the environment level, the same resource template can be created for multiple environments. This allows you to create new BPM Application instances in new as well as existing installations.

Deploying the application multiple times on nodes belonging to the same environment

As long as the BPM application instances are distributed on exclusive sets of nodes, the same BPM Application can be instantiated multiple times in the same environment.

Hibernate

The Hibernate resource template represents a [Hibernate](#) resource. Used by component implementations to access databases, the hibernate is a framework that supports storing Java objects in a relational database. Hibernate solves object-relational impedance mismatch by replacing direct database access with high-level, object-handling functions.

General

Property	Required ?	Editable?	Accept s SVars?	Description
Data Source	Y	Y	N	The name of a JDBC resource that represents the connection to the database.

Property	Required ?	Editable?	Accept s SVars?	Description
Schema Generation Type	N	N	N	<p>Indicate whether to create or validate the schema in the database when the session factory is created:</p> <ul style="list-style-type: none"> • Do Nothing - Indicate that only data is added, changed, and deleted. If the schema does not already exist, the application will experience errors when it runs. • Validate - Validate the schema. • Create - Create the schema every time the session factory is created, deleting old schema and data if it exists. • Create Drop - Same as Create, but drops the schema at the end of the session. • Update - Update the schema with the changes implied by the Java objects being mapped to the database. <p>Default: Do Nothing.</p>

Property	Required ?	Editable?	Accepts SVars?	Description
Dialect	Y	Y	Y	<p>The class name of a Hibernate dialect that enables Hibernate to generate SQL optimized for a particular relational database. The supported dialects are:</p> <ul style="list-style-type: none"> org.hibernate.dialect <ul style="list-style-type: none"> DB2390Dialect DB2400Dialect DB2Dialect FirebirdDialect FrontbaseDialect HSQLDialect InformixDialect IngresDialect InterbaseDialect MckoiDialect MySQLDialect MySQLInnoDBDialect MySQLMyISAMDialect Oracle9Dialect OracleDialect PointbaseDialect PostgreSQLDialect ProgressDialect SAPDBDialect SQLServerDialect SybaseAnywhereDialect SybaseDialect com.tibco.amf.sharedresource.runtime.core.hibernate.dialects <ul style="list-style-type: none"> DB2Dialect HSQLDialect MySQL5Dialect Oracle9Dialect Oracle10GDialect SQLServerDialect

Property	Required?	Editable?	Accepts SVars?	Description
				Default: com.tibco.amf.sharedresource.runtime.core. hibernate.dialects.HSQLDialect

Advanced

Property	Required?	Editable?	Accepts SVars?	Description
Enable SQL Logging	N	N	Y	Permit data collection in the SQL Server transaction log file. Default: Unchecked.
Batch Size	N	Y	Y	Enables JDBC batch processing. Default: 5.
Share Session Factory	N	Y	Y	Indicate whether clients share the session factory or whether a new factory is created for each client. Default: Checked.


Property	Required ?	Editable?	Accepts SVars?	Description
Properties	N	Y	N	<p>Hibernate configuration properties:</p> <ul style="list-style-type: none"> • Format SQL Enabled • Default Schema • Default Catalog • Max Fetch Depth • Default Batch Fetch Size • Use Order Updates • Use Order Inserts • Use Generate Statistics • Use Identifier Rollback • Use SQL Comments • Fetch Size • Batch Versioned Data • Batch Factory Class • Use Scrollable Resultset • Use Stream For Binary • Use Get Generated Keys • Connection Isolation • Use Auto Commit • Connection Release Mode • Cache Provider Class • Use Minimal Puts • Use Query Cache • Use Second Level Cache • Query Cache Factory • Cache Region Prefix • Use Structured Entries • Transaction Factory Class • JTA Transaction JNDI Name • Flush Before Completion • Auto Close Session • Query Factory Class • Query Substitutions • Use Reflection Optimizer

HTTP Client

The HTTP Client resource template represents an outgoing HTTP connection. HTTP clients are used by a reference's SOAP binding.

General

Property	Required?	Editable ?	Accepts SVars?	Description
Machine Name	Y	Y	Y	The name of the host that accepts the incoming requests. For machines that have only one network card, the default value localhost specifies the current machine. For machines that have more than one network card, this field specifies the host name of the card that used to accept incoming HTTP requests. Default: localhost.
Port	Y	Y	Y	The port number on which to invoke outgoing HTTP requests. Default: 80.
Idle Timeout (s)	N	Y	Y	The length of time to wait before closing an inactive connection. If more than zero, and data transmission has not finished, a call to close the connection blocks the calling program until the data is transmitted or until the specified timeout occurs. If 0, a call to close the connection returns without blocking the caller and an attempt is made to send the data. Normally this transfer is successful, but it cannot be guaranteed. This value should be changed only on the advise of TIBCO Support. Default: 0 s.
Socket Timeout (ms)	N	Y	Y	Defines the socket timeout (SO_TIMEOUT), which is the timeout for waiting for data or a maximum period inactivity between consecutive data packets. This should be changed when connecting to very slow external services. A timeout value of zero is interpreted as an infinite timeout. Default: 0 ms.

Property	Required?	Editable?	Accepts SVars?	Description
Connection Timeout (ms)	N	Y	Y	<p>Determines the timeout until a connection is established. This should be changed when connecting to very slow external services. A timeout value of zero is interpreted as an infinite timeout.</p> <div>  <p>The timeout is influenced by operating system specific behavior at the TCP socket layer. On Windows 2008, Windows 7 and Windows XP the timeout value configured in this field is not honored, and instead it uses an internal timeout of around 21 seconds. Some versions of Linux, such as Ubuntu, also do not honor this timeout.</p> </div> <p>Default: 0 ms.</p>

SSL

Property	Required?	Editable?	Accepts SVars?	Description
Enable SSL	Y	Y	N	<p>Enable SSL connections. When checked, the SSL properties display.</p> <p>Default: Unchecked.</p>
SSL Client Provider	Y	Y	N	The name of an SSL Client Provider resource.
Configure SSL	N	N	N	<p>(Not applicable to some resource templates) Invokes a wizard to import certificates from an SSL-enabled server, optionally create an SSL Client Provider resource, and configure the trust store of the newly created or an existing SSL Client Provider with the imported certificates. When you complete the wizard, the SSL Client Provider field is filled in.</p>

Advanced Configuration

Property	Required ?	Editable?	Accepts SVars?	Description
Accept Redirect	N	N	N	<p>Indicates whether the HTTP method should automatically follow HTTP redirects.</p> <p>This option is used when client connection receives the redirect responses from server like Moved Permanently, Moved Temporarily, Temporary Redirect and so on.</p> <p>Default: Unchecked.</p>
Reuse Address	N	N	N	<p>When a TCP connection is closed, the connection might remain in a timeout state for a period of time after the connection is closed (typically known as the TIME_WAIT state or 2MSL wait state).</p> <p>For applications using a well-known socket address or port, it might not be possible to bind a socket to the required SocketAddress if there is a connection in the timeout state involving the socket address or port.</p> <p>Default: Unchecked.</p>
Disable Connection Pooling	N	N	N	<p>Indicate whether to use the single or multi-threaded connection manager.</p> <p>Default: Unchecked.</p>
Suppress TCP Delay	N	N	N	<p>Determines whether the Nagle algorithm is used.</p> <p>The Nagle algorithm tries to conserve bandwidth by minimizing the number of segments that are sent. When applications wish to decrease network latency and increase performance, they can disable Nagle's algorithm by enabling Suppress TCP Delay.</p> <p>Data will be sent earlier at the cost of an increase in bandwidth consumption and the number of packets.</p> <p>Default: Checked.</p>
Stale Check	N	N	N	<p>Determines whether the stale connection check is to be used. Disabling the stale connection check can result in slight performance improvement at the risk of getting an I/O error when executing a request over a connection that has been closed at the server side.</p> <p>Default: Unchecked.</p>

Property	Required ?	Editable?	Accepts SVars?	Description
Buffer Size (B)	N	Y	N	<p>Socket buffer size in bytes.</p> <p>A suggestion to the kernel from the application about the size of the buffers to use for the data transferred over the socket.</p> <p>Default: -1. Allow the runtime to determine the buffer size.</p>
Connection Retrieval Timeout (ms)	N	Y	Y	<p>The timeout, in milliseconds, until a connection is established.</p> <p>Default: 0.</p>
Local Socket Address	N	Y	N	<p>Local host address to be used for creating the socket.</p> <p>Default: None.</p>
Maximum Total Connections	N	Y	Y	<p>Controls the maximum number of simultaneous active connection that this resource instance allows. The value should be increased for application that creates a lot of long-lived connections.</p> <p>Default: 20.</p>
Maximum Total Connections per Host	N	Y	Y	<p>Controls the maximum number of simultaneous active connection to a same host that this resource instance allows. This number cannot be greater than Maximum Total Connections.</p> <p>Default: 2.</p>

HTTP Proxy

Property	Required ?	Editable?	Accepts SVars?	Description
Configure Proxy	N	N	N	<p>Check the check box to configure the HTTP Proxy options described in this table.</p> <p>Default: Unchecked</p>
Proxy Type	Y	N	N	<p>Type of proxy server. You can select HTTP or SOCKS V4 / V5.</p> <p>Default: HTTP</p>
Proxy Host	Y	Y	Y	<p>Address of the proxy host.</p> <p>Default: localhost</p>

Property	Required ?	Editable?	Accepts SVars?	Description
Proxy Port	Y	Y	Y	Port of the proxy host. Default: 8080
Configure BASIC authentication	N	Y	N	Check the box to configure access to proxy server with a username and password. Default: Unchecked When you check this check box, the fields for specifying the username and password are enabled. Default username: username Default password: None

JDBC

The JDBC resource template represents a JDBC connection that is used by component implementations to access databases.

General

Property	Required?	Editable?	Accept SVars?	Description
Connection Type	Y	N	N	<p>The type of the JDBC connection:</p> <ul style="list-style-type: none"> • Direct The connection to the database is through a vendor-specific driver. When selected, the Database Driver and Database URL fields display. • XA The connection to the database is through a vendor-specific data source. When selected, the Data Source field displays. A component implementation that uses a JDBC connection of connection type XA typically executes within a global transaction and consequently may not explicitly commit transactions. To ensure that such implementations always behave correctly, the TIBCO ActiveMatrix platform detects when such a resource is used outside of a global transaction and enables the JDBC autocommit feature, so that all database access by the component is committed. Default Login Timeout: 60000 ms (60s) <p>Default: Direct</p>

Direct

Property	Required?	Editable?	Accepts SVars?	Description
Database Driver	Y	Y	Y	<p>The name of the JDBC driver class. You can select from a drop-down list of supported drivers or type the name of a custom driver:</p> <ul style="list-style-type: none"> • org.hsqldb.jdbcDriver • com.microsoft.sqlserver.jdbc.SQLServerDriver • com.mysql.jdbc.Driver • oracle.jdbc.OracleDriver • com.ibm.db2.jcc.DB2Driver • org.postgresql.Driver <p>Additional drivers available when using TIBCO Business Studio:</p> <ul style="list-style-type: none"> • com.ibm.as400.access.AS400JDBCdriver • com.informix.jdbc.IfxDriver • ca.edbc.jdbc.EdbcDriver <p>When you select a driver, the Database URL field is populated with a template for the URL of the driver.</p> <p>Default: org.hsqldb.jdbcDriver.</p>

Property	Required?	Editable?	Accepts SVars?	Description
Database URL	Y	Y	Y	<p>The URL to use to connect to the database. A template of the URL is supplied for the driver you select in the Database Driver field or you can type the name of a URL:</p> <ul style="list-style-type: none"> • jdbc:hsqldb:hsq://localhost:<port#>/<db_instancename> • jdbc:sqlserver://<server Name>:<portNumber>;databaseName=<dbname>; • jdbc:mysql://<localhost>:<port>/<DBName> • jdbc:oracle:thin:@<machine_name>:<port>:<instance_name> • jdbc:db2://<host>:<port default is 50000>/<database name> • jdbc:postgresql://<servername>:<portnumber>/<dbname> <p>Available when using TIBCO Business Studio:</p> <ul style="list-style-type: none"> • jdbc:as400://server<server_ip>;libraries=<lib> • jdbc:informix-sqli://<host>:<port >/<database>;informixserver=<server> • jdbc:edbc://<host>:<port>/<database> <p>You must supply the portions of the URL shown between angle brackets and remove the angle brackets.</p> <p>Default: jdbc:hsqldb:hsq://localhost:<port#>/<db_instance name>.</p>

XA

Property	Required?	Editable?	Accept SVars?	Description
Data Source	Y	Y	Y	<p>The fully-qualified name of the javax.sql.XADataSource implementation class. The supported classes are:</p> <ul style="list-style-type: none"> com.ibm.db2.jcc.DB2XADataSource com.mysql.jdbc.jdbc2.optional.MysqlXADataSource oracle.jdbc.xa.client.OracleXADataSource com.microsoft.sqlserver.jdbc.SQLServerXADataSource org.postgresql.xa.PGXADataSource <p>Default: oracle.jdbc.xa.client.OracleXADataSource</p>

Property	Required?	Editable?	Accepts SVars?	Description
Maximum Connections	N	Y	Y	<p>The maximum number of database connections to allocate. The minimum value that can be specified is 0.</p> <p>Default: 10.</p>
Login Timeout (ms)	N	Y	Y	<p>Time to wait for a successful database connection. If the JDBC driver does not support connection timeouts, the value of this field is ignored. Only JDBC drivers that support connection timeouts use this configuration field. Most JDBC drivers support connection timeouts.</p> <p>Default: 60000 (60 seconds).</p>
Supports Transactions	N	Y	Y	<p>Indicate whether the application demarcates transaction boundaries. If unchecked, the application does not demarcate transaction boundaries and all SQL statements are autocommitted.</p> <p>If checked, the application demarcates transaction boundaries.</p> <p>Default: Unchecked.</p>

Login Credentials

Property	Required?	Editable?	Accepts SVars?	Description
Login Credentials	Y	Y	N	<p>Indicate how the credentials required to authenticate to a server are provided:</p> <ul style="list-style-type: none"> • Identity Provider - Provide username and password credentials encapsulated in an identity provider resource. When selected, the Identity Provider field is activated. • Username + Password - Provide inline username and password credentials. When selected, the Username and Password fields are activated. <p>Default: Identity Provider</p>
Identity Provider	N	Y	N	Name of the Identity Provider resource used to authenticate the user.
Username	N	Y	N	Username used to authenticate connections to the server.
Password	N	Y	N	<p>User's password used to authenticate connections to the server.</p> <p>(Administrator only) For superusers, passwords display encrypted. For non-superusers, the password doesn't display even if it was set when it was created. If you have permission to edit the password, you can specify a new value and save. If you edit other fields, the old value for the password field is retained. If you want to set an empty value as password, click the link Set Blank Password.</p>

SSL

GUI	Editable ?	Required?	Accepts SVars?	Description
Enable SSL	N	N	N	<p>Enable SSL connections. When checked, the SSL properties display.</p> <p>Default: Unchecked.</p>

GUI	Editable ?	Required?	Accepts SVars?	Description
SSL Client Provider	N	Y	Y	The name of an SSL Client Provider resource instance.
Configure SSL	N	N	N	(Administrator only) Invokes a wizard to import certificates from an SSL-enabled server, optionally create an SSL Client Provider resource instance, and configure the trust store of the newly created or an existing SSL Client Provider with the imported certificates. When you complete the wizard, the SSL Client Provider field is filled in.

Advanced

GUI	Editable ?	Required?	Accepts SVars?	Description
Host Type Properties	Y	N	N	Properties to configure the connection between the JDBC resource and a specific type of host.
commitBeforeAutocommit	Y	N	N	Indicates whether the driver requires a commit to be performed before enabling auto-commit on a connection. This should be (and is, by default) set to false for compliant drivers to avoid extraneous commits to the database. Default: false.

GUI	Editable ?	Required?	Accepts SVars?	Description
exceptionSorterClass	Y	N	N	<p>The class used by the resource adapter to judge if an exception is fatal to the connection. That is, whether the connection pool should discard the connection from the pool, since it is no longer reusable. As the name implies, the default <code>SQLState08ExceptionsAreFatalSorter</code> treats SQL State 8 exceptions as fatal (connection errors). All other exceptions do not result in any connection pool action (but of course are passed up to the application for it to react as it wishes). The class must implement <code>org.tranql.connector.ExceptionSorter</code>.</p> <p>Default: <code>com.tibco.amf.sharedresource.runtime.tibcohost.jdbc.SQLState08ExceptionsAreFatalSorter</code>.</p>
POOL_MIN_SIZE	Y	N	N	<p>Minimum number of connections in the pool.</p> <p>Default: 5.</p>
POOL_BLOCKING_TIMEOUT (ms)	Y	N	N	<p>The length of time a requestor will wait for a connection when the pool is at maximum.</p> <p>Default: 60000 ms.</p>
POOL_IDLE_TIMEOUT (min)	Y	N	N	<p>The length of time after which idle connections are closed.</p> <p>Default: 5 min.</p>
preparedStatementCacheSize	Y	N	N	<p>The size of the cache containing prepared statements. The size should correspond to the number of JDBC statements you expect your application to reuse.</p> <p>Default: 0; that is, the cache is disabled.</p>

Direct

Property	Required?	Editable?	Accepts SVars?	Description
Connection Properties	N	Y	N	Properties to configure connections to a database driver. The properties are vendor specific.

XA

Property	Required ?	Editable?	Accept SVars?	Description
Connection Properties	N	Y	N	Properties to configure connections to a data source. The properties are vendor specific.

Configuring Third-Party JDBC Drivers

Before deploying an application that uses a third-party JDBC server, the JDBC client library must be packaged and installed on each node on which the application will run.

When you add a DAA containing a client library to a run configuration, TIBCO Business Studio deploys the DAA to the local debug node.

Procedure

1. Package the database driver in a [DAA](#).
2. Add the DAA to the [run configuration](#).

JMS Resource Templates

JMS resource templates enable applications to access objects maintained in JMS servers.

The JMS resource templates are:

[JNDI Connection Configuration](#) - Provides a JNDI connection to look up a JMS server.

[JMS Connection Factory](#) - Used to create an outbound connection to a JMS server.

[JMS Destination](#) - Used for Request/Reply messages. Specifies destination objects, which represent virtual channels (topics and queues) in JMS. When a message is sent, it is addressed to a destination, not to a specific application. Any application that subscribes or registers an interest in that destination can receive that message. Depending on the JMS messaging model used, the destination is called a topic or a queue. In the publish-subscribe model, a message is published for many subscribers to a topic (destination). In the point-to-point model, one message is sent to one potential receiver using a queue (destination).

JMS Resource Template Relationships

The JMS resource templates are used in different combinations to accomplish the tasks involved in setting up JMS enterprise messaging:

- Identifying the JMS server to connect to
- Establishing request communication
- Establishing reply communication

Identifying the JMS server is accomplished through the JNDI Connection resource template. All the other JMS resource templates contain a link for the JNDI Connection that assists them in determining which JMS server to look up. Additionally, before the connection to the JNDI server is made, the JNDI might require authentication. Authentication can take the form of a username and password, or supplying credential information stored in a keystore using an identity provider. If the JNDI server is SSL-enabled, you provide the required SSL configuration.

To establish request or reply communication, you need these resource templates: JMS Connection Factory, JMS Destination, and JNDI Connection.



Only JMS Connection Factory resource template is needed, if direct destinations are used.

Configuring Third-Party JMS Drivers

Before you can deploy an application that uses a third-party JMS server, the JMS client library must be packaged and installed on each node on which the application will run.

When you add a DAA containing a client library to a run configuration, TIBCO Business Studio deploys the DAA to the local debug node.

Procedure


1. Package the third-party JMS client driver in a DAA. See [Creating a Library Plug-in Project](#).
2. Add the DAA to the run configuration. See [Creating a Run Configuration](#).

JMS Connection Factory

A JMS Connection Factory creates an outbound connection to a JMS server.

Property	Editable ?	Required ?	Accept s SVars?	Description
Connection Factory JNDI Name	Y	Y	Y	JNDI name of the JMS Connection Factory that points to a particular queue or topic.

Property	Editable ?	Required ?	Accepts SVars?	Description
Maximum Pool Size	Y	Y	Y	<p>(Optional) This property is available when creating a new JMS Connection Factory Resource Template. It can also be updated for an existing JMS Connection Factory Resource Template.</p> <p>If no value is specified, the default value for Maximum Pool Size is 20. You can override the default value using the property <code>com.tibco.amf.sharedresource.jms.connection.pool.maxsize</code> in <code>SystemNode.tra</code>. The value can be specified in the TRA file as follows:</p> <pre>java.property.com.tibco.amf.sharedresource.jms.connection.pool.maxsize=<new_value></pre> <p>and also using the TIBCO ActiveMatrix Administrator UI:</p> <ul style="list-style-type: none"> property: <code>com.tibco.amf.sharedresource.jms.connection.pool.maxsize</code> value: <code><new_value></code> <p>When you change the Maximum Pool Size of an existing Resource Template, the UI prompts you to reinstall all Resource Instances of that Resource Template. After the Resource Instances are re-installed, the new value of Maximum Pool Size is applied to runtime.</p> <p>The SystemNode should be restarted after setting the Maximum Pool Size property.</p> <p>NOTE: While creating a resource template, the value provided using the UI or CLI (via attribute <code>maxPoolSize</code>) takes precedence over the value specified in the SystemNode TRA file. The <code>maxPoolSize</code> attribute can be specified as follows. Refer to the <code><TIBCO_HOME>/administrator/<version_number>/samples/jmssr_data.xml</code> configuration file for details.</p> <pre><ResourceTemplate xsi:type="amxdata_jmssr:JNDIConnectionFactoryResourceTemplate" name="NewJndiConnectionFactorySharedResource" jndiName="GenericConnectionFactory" jndiConnectionConfigurationName="NewJndiConnectionConfigurationSharedResource"></pre>

Property	Editable ?	Required ?	Accepts SVars?	Description
				e" description="JndiConnectionFactory" xa="false" maxPoolSize="20">
JNDI Connection Configuration	Y	Y	N	<p>The name of a JNDI Connection Configuration resource.</p> <p> You can use a substitution variable for JNDI connection configuration in the JMS Connection Factory resource template in TIBCO ActiveMatrix Administrator.</p>

Security

Property	Required?	Editable ?	Accepts SVars?	Description
Enable Authentication	N	N	N	<p>(Administrator UI only) Enable server authentication. When checked, the authentication properties: Login Credentials, Username, and Password are displayed. The Enable Authentication property is only available in the Administrator UI.</p> <p>Default: Unchecked.</p>
Login Credentials	Y	Y	N	<p>Indicate how the credentials required to authenticate to a server are provided:</p> <ul style="list-style-type: none"> • None • Username + Password - Provide inline username and password credentials. When selected, the Username and Password fields are activated. • Identity Provider - Provide username and password credentials encapsulated in an identity provider resource. When selected, the Identity Provider field is activated. <p>Default: None</p>
Username	N	Y	N	Username used to authenticate connections to the server.

Property	Required?	Editable ?	Accepts SVars?	Description
Password	N	Y	N	User's password used to authenticate connections to the server. (Administrator only) For superusers, passwords display encrypted. For non-superusers, the password doesn't display even if it was set when it was created. If you have permission to edit the password, you can specify a new value and save. If you edit other fields, the old value for the password field is retained. If you want to set an empty value as password, click the link Set Blank Password .
Identity Provider	N	Y	N	Name of the Identity Provider resource used to authenticate the user.
Enable SSL	Y	Y	N	Enable SSL connections. When checked, the SSL properties display. Default: Unchecked.
SSL Client Provider	Y	Y	N	The name of an SSL Client Provider resource.

JMS Destination

A JMS Destination resource template specifies destination objects, which represent virtual channels (topics and queues) in JMS. It is used for Request/Reply messages.

When a message is sent, it is addressed to a destination, not to a specific application. Any application that subscribes or registers an interest in that destination can receive that message. Depending on the JMS messaging model used, the destination is called a topic or a queue. In the publish-subscribe model, a message is published for many subscribers to a topic (destination). In the point-to-point model, one message is sent to one potential receiver using a queue (destination).

Property	Editable ?	Required ?	Accepts SVars?	Description
Destination JNDI Name	Y	Y	Y	A JNDI name of a JMS destination that points to a particular queue or topic.
JNDI Connection Configuration	Y	Y	N	The name of a JNDI Connection Configuration.

JNDI Connection Configuration

A JNDI Connection Configuration resource template provides a JNDI connection to look up a JMS server.

General

Property	Editable ?	Required?	Accepts SVars?	Description
JNDI Provider	Y	Y	N	<p>The provider to use for JNDI lookup:</p> <ul style="list-style-type: none"> • TIBCO EMS • Progress SonicMQ • IBM MQ • Custom - Used for custom JNDI providers. <p>The Initial Context Factory field is populated based on the JNDI provider selected. SSL lookup is only available for the TIBCO EMS provider.</p> <p>Default: TIBCO EMS.</p>
Initial Context Factory	Y	Y	Y	<p>Initial context factory to be used for the JNDI lookup. The value for Initial Context Factory is set based on the JNDI provider selected:</p> <ul style="list-style-type: none"> • TIBCO EMS - <code>com.tibco.tibjms.naming.TibjmsInitialContextFactory</code> value is populated • Progress SonicMQ - the <code>com.sonicsw.jndi.mfcontext.MFContextFactory</code> value is populated • IBM MQ - <ul style="list-style-type: none"> – <code>com.sun.jndi.ldap.LdapCtxFactory</code> for the JNDI lookup in LDAP. Pair this value with the Naming Provider URL: <code>ldap://<ldap_url></code>. – <code>com.sun.jndi.fscontext.RefFSContextFactory</code> for the JNDI lookup in a file system. Pair this value with the Naming Provider URL: <code>file:<url_of_bindings_file></code>. • Custom provider - Specify the custom initial context factory value. <p>Default: <code>com.tibco.tibjms.naming.TibjmsInitialContextFactory</code></p>

Property	Editable ?	Required?	Accepts SVars?	Description
Provider URL	Y	Y	Y	<p>Provider URL of the JNDI server. The value for Naming Provider URL is set based on the JNDI provider selected:</p> <ul style="list-style-type: none"> • TIBCO EMS - <code>tibjmsnaming://<host>:<port></code> • Progress SonicMQ - <code>tcp://<host>:<port></code> • IBM MQ - <ul style="list-style-type: none"> – <code>ldap://<ldap_url></code> for the JNDI lookup in LDAP. Example: <code>ldap://mymachine.tibco.com:2076/dc=tibco,dc=com</code>. Pair this value with the Initial Context Factory: <code>com.sun.jndi.ldap.LdapCtxFactory</code>. – <code>file:<url_of_bindings_file></code> for the JNDI lookup in a file system. Example: <code>file:/D:/Program Files/IBM/fileBinding</code>. Pair this value with the Initial Context Factory: <code>com.sun.jndi.fscontext.RefFSContextFactory</code>. • Custom - specify the custom provider URL. <p>Default: <code>tibjmsnaming://<host>:<port></code>.</p> <p>The Naming Provider URL is validated using recommendation of the "Uniform Resource Identifiers (URI): Generic Syntax" [RFC2396] standard for the TIBCO EMS, Progress SonicMQ and IBM MQ JNDI provider.</p>

Security

Property	Required?	Editable ?	Accepts SVars?	Description
Enable Authentication	N	N	N	<p>(Administrator UI only) Enable server authentication. When checked, the authentication properties: Login Credentials, Username, and Password are displayed. The Enable Authentication property is only available in the Administrator UI.</p> <p>Default: Unchecked.</p>

Property	Required?	Editable ?	Accepts SVars?	Description
Login Credentials	Y	Y	N	<p>Indicate how the credentials required to authenticate to a server are provided:</p> <ul style="list-style-type: none"> • None • Username + Password - Provide inline username and password credentials. When selected, the Username and Password fields are activated. • Identity Provider - Provide username and password credentials encapsulated in an identity provider resource. When selected, the Identity Provider field is activated. <p>Default: None</p>
Username	N	Y	N	Username used to authenticate connections to the server.
Password	N	Y	N	<p>User's password used to authenticate connections to the server.</p> <p>(Administrator only) For superusers, passwords display encrypted. For non-superusers, the password doesn't display even if it was set when it was created. If you have permission to edit the password, you can specify a new value and save. If you edit other fields, the old value for the password field is retained. If you want to set an empty value as password, click the link Set Blank Password.</p>
Identity Provider	N	Y	N	Name of the Identity Provider resource used to authenticate the user.
Enable SSL	Y	Y	N	<p>Enable SSL connections. When checked, the SSL properties display.</p> <p>Default: Unchecked.</p>
SSL Client Provider	Y	Y	N	The name of an SSL Client Provider resource.

Advanced

A list of properties used for JNDI lookup.

Application Properties

Property	Description
Name	Name of the property.
Type	Type of the property. One of: string, boolean, byte, short, char, int, long, float, or double.
Value	Property value. Default: Depends on value of 'Type'.


You can set a property value to a literal or a substitution variable.

JMS Destination Configuration



This Resource Template is deprecated. Any use of this Resource Template should be avoided as it may be removed in a future release.

A JMS Destination Configuration resource template specifies what topic or queue to listen to for request messages.

Property	Editable?	Required?	Accepts SVars?	Description
Destination JNDI Name	Y	Y	Y	A JNDI name of a JMS destination that points to a particular queue or topic.
JNDI Connection Configuration	Y	Y	N	<p>The name of a JNDI Connection Configuration resource.</p> <p> You can use a substitution variable for JNDI connection configuration in the JMS Connection Factory resource template in TIBCO ActiveMatrix Administrator.</p>

LDAP Connection

An LDAP Connection resource template represents a connection to an LDAP server. Used by component implementations to look up names in an LDAP directory server.

General

Property	Required?	Editable ?	Accepts SVars?	Description
Connection Factory	Y	Y	Y	<p>The factory object that provides the starting point for resolution of names within the LDAP server.</p> <p>Default: com.sun.jndi.ldap.LdapCtxFactory.</p>

Property	Required?	Editable?	Accepts SVars?	Description
Provider URL	Y	Y	Y	<p>The URL that provides the host and port number on which the LDAP server is listening for connections. It can also include a Base DN, the DN of an entry in the directory.</p> <p>The Base DN:</p> <ul style="list-style-type: none"> Identifies the LDAP entry that is the starting point of all searches Limits the searches to a subtree of the LDAP Server's directory <p>If the Base DN is not specified, all searches begin at the root DN.</p> <p>Any unsafe characters in the URL must be represented by a special sequence of characters called escaping. For example, a space must be represented as %20. Thus, the DN <code>ou=Product Development</code> must be encoded as <code>ou=Product%20Development</code>.</p> <p>Default: <code>ldap://localhost:389</code>.</p>
Connection Timeout (ms)	N	Y	Y	<p>The time to wait for a response from the LDAP directory server.</p> <p>Default: 0.</p>

Login Credentials

Property	Required?	Editable?	Accepts SVars?	Description
Login Credentials	Y	Y	N	<p>Indicate how the credentials required to authenticate to a server are provided:</p> <ul style="list-style-type: none"> Identity Provider - Provide username and password credentials encapsulated in an identity provider resource. When selected, the Identity Provider field is activated. Username + Password - Provide inline username and password credentials. When selected, the Username and Password fields are activated. <p>Default: Identity Provider</p>

Property	Required?	Editable?	Accepts SVars?	Description
Identity Provider	N	Y	N	Name of the Identity Provider resource used to authenticate the user.
Username	N	Y	N	Username used to authenticate connections to the server.
Password	N	Y	N	<p>User's password used to authenticate connections to the server.</p> <p>(Administrator only) For superusers, passwords display encrypted. For non-superusers, the password doesn't display even if it was set when it was created. If you have permission to edit the password, you can specify a new value and save. If you edit other fields, the old value for the password field is retained. If you want to set an empty value as password, click the link Set Blank Password.</p>

Advanced

Property	Required?	Editable ?	Accept SVars?	Description
Pool Size	N	Y	Y	<p>The preferred number of connections per connection identity that should be maintained concurrently.</p> <p>Default: 10.</p>
Pool Maximum	N	Y	Y	<p>The maximum number of connections per connection identity that can be maintained concurrently.</p> <p>Default: 15.</p>
Pool Initial	N	Y	Y	<p>The number of connections per connection identity to create when initially creating a connection for the identity.</p> <p>Default: 5.</p>
Pool Timeout (ms)	N	Y	Y	<p>The length of time that an idle connection may remain in the pool without being closed and removed from the pool.</p> <p>Default: 300000.</p>

Property	Required?	Editable?	Accept SVars?	Description
Follow Referrals	N	N	Y	Indicate whether an LDAP server should return a reference (a referral) to another LDAP server which may contain further information instead of returning a result. Default: Unchecked.

SSL

Property	Required?	Editable?	Accepts SVars?	Description
Enable SSL	Y	Y	N	Enable SSL connections. When checked, the SSL properties display. Default: Unchecked.
SSL Client Provider	Y	Y	N	The name of an SSL Client Provider resource.

Security Resource Templates

Security features are provided by a set of resource templates that provide access to various types of security providers: identity, trust, mutual identity, keystore, SSL client and server, and authentication.

Identity, keystore, trust, and mutual identity providers enable clients and servers to assert and establish identity. SSL resource templates are used to enable SSL configurations for use in resource templates that define connections to various types of servers. For example, the SSL configuration for an HTTP Client, is set by an SSL Client Provider. The SSL Client Provider in turn references a Keystore Provider to establish the identity of a trusted server. Authentication providers enable connections to authentication services. Some resource templates types, for example authentication providers, are only available in Administrator.

Type	Resource Template
Identity	<ul style="list-style-type: none"> Identity Provider - The Identity Provider resource template provides access to a username and password credential stored in a keystore. Kerberos Identity Provider - The Kerberos Identity Provider resource template provides access to an identity stored in a Kerberos authentication server. Keystore Provider - The Keystore Provider resource template provides access to a keystore.
SSL	<ul style="list-style-type: none"> SSL Client Provider - The SSL Client Provider resource template maintains the credentials required by an SSL client.

Identity Provider

The Identity Provider resource template provides access to a username and password credential stored in a keystore.

General

Property	Required?	Editable ?	Accepts SVars?	Description
Keystore Provider to Supply Identity	Y	Y	N	Name of a Keystore Provider resource that maintains a keystore used to assert an identity.
Enable Access to Credential Store Containing Identity (optional)	N	N	N	Enables access to an identity keystore. To establish SSL connections, certain third-party systems such as MySQL require access to a keystore file location. In such situations Administrator provides a copy of credentials in a keystore, which are then written to disk and used by the third party as the SSL credential store. To prevent Administrator from providing credentials, uncheck the checkbox. Default: Unchecked.
Key Alias to Access Identity	Y	Y	N	Name of the alias used to access the identity.
Key Alias Password	Y	Y	N	Password for the alias. (Administrator only) For superusers, passwords display encrypted. For non-superusers, the password doesn't display even if it was set when it was created. If you have permission to edit the password, you can specify a new value and save. If you edit other fields, the old value for the password field is retained. If you want to set an empty value as password, click the link Set Blank Password .

Kerberos Identity Provider

The Kerberos Identity Provider resource template provides access to an identity stored in a [Kerberos](#) authentication server.

Property	Editable ?	Required?	Accepts SVars?	Description
Kerberos Authentication Provider	Y	Y	Y	The name of a Kerberos Authentication Provider containing the identity. Default: None.
Kerberos Service Principal Name	Y	N	Y	The name of a Kerberos service principal. Default: None.
Kerberos Client Principal Name	Y	N	Y	The name of a Kerberos client principal. Specify this information to gain access to the private key of the client principal. Default: None.
Kerberos Client Principal Password	Y	N	N	The password of the Kerberos client principal. In addition to the Kerberos Client Principal Name, specify this information to gain access to the private key of the client principal. Default: None.



Kerberos Identity Provider must be set up before configuring WSS Authentication.

Keystore Provider

The Keystore Provider resource template provides access to a keystore.

General

Property	Required ?	Editable?	Accepts SVars?	Description
Keystore Served From	Y	Y	N	Location of the keystore: <ul style="list-style-type: none"> Store the keystore in Administrator and serve it from here The keystore is hosted externally at URL
Administrator - Upload Keystore From	Y	Y	N	Path to the keystore to be uploaded into Administrator. After the keystore is uploaded, a link displays from which the keystore can be downloaded.

Property	Required ?	Editable?	Accept s SVars?	Description
URL	Y	Y	Y	Location of the external keystore.
Password	Y	Y	Y	<p>Password for the keystore.</p> <p>(Administrator only) For superusers, passwords display encrypted. For non-superusers, the password doesn't display even if it was set when it was created. If you have permission to edit the password, you can specify a new value and save. If you edit other fields, the old value for the password field is retained. If you want to set an empty value as password, click the link Set Blank Password.</p>
Provider	N	Y	Y	<p>Name of the keystore provider:</p> <ul style="list-style-type: none"> • SunJCE (JCEKS format) • SUN (JKS format) • IBMJCE (IBM JREs) • SunJSSE (PKCS12 format) <p>Default: Empty. The first matching provider supporting the format will be chosen.</p>
Type	Y	Y	Y	<p>Type of the keystore: JCEKS, JKS, PKCS12.</p> <p>Default: JKS.</p>
Refresh Interval (ms)	Y	Y	Y	<p>Refresh interval, greater than 0. If the keystore provider is accessed after the refresh interval has expired:</p> <ol style="list-style-type: none"> 1. The keystore provider is refreshed from its backing keystore. 2. The refresh timer is reset to zero. 3. Operations on the keystore provider are performed on the refreshed copy. <p>Default: 3600000.</p>

Maximum Pool Size Parameter in CLI

You can now set the `maxPoolSize` value in the `resourcetemplate_data.xml` to control the Pool Size. This file is located in the `<CONFIG_HOME>\admin\<InstanceName>\samples` folder.

For example, in the `resourcetemplate_data.xml` excerpt shown below, the `maxPoolSize` is set to 45:

```
<ResourceTemplate xsi:type="amxdata:KeystoreCspResourceTemplate"
name="KeystoreRT"
description="This is a Keystore RT"
keyStoreLocation="/path/to/keystore.jceks"
```

```
keyStorePassword="unique"
keyStoreType="JCEKS"
keyStoreProvider="SunJCE"
maxPoolSize="45"
keyStoreRefreshInterval="3600000"/>
```



The above CLI Data file changes are applicable to Resource Templates with a 'Global' scope. If a given Resource Template is scoped at 'Application' or 'Environment' level, `resourcetemplate_scope_build.xml` and `resourcetemplate_scope_data.xml` files will have to be updated.

Keystores

If you set up your environment for SSL, you have to set up a keystore. As part of the process, you configure a keystore provider.

SSL uses keys and certificates when it establishes the secure connection. A *keystore* is a database of keys and certificates. A keystore password is required to access or modify the keystore.

Access to keystores is provided by a Keystore Provider resource instance. Keystores can be stored internally in Administrator or externally.

ActiveMatrix Administrator Default Keystore

In TIBCO ActiveMatrix access to keystores is provided by a Keystore Provider resource instance. When you create an Administrator server, TIBCO ActiveMatrix includes a default keystore provider resource template named `tibco.admin.default.keystore` that references the default keystore `CONFIG_HOME/admin/amxadmin/shared/repo/trunk/artifacts/keystore/admin_default_keystore.jceks`.

Keystore Entries

A keystore has two types of entries:

- Private key - holds a cryptographic private key, which is optionally stored in a protected format to prevent unauthorized access. The private key is accompanied by a certificate chain for the corresponding public key. Private keys and certificate chains are used by a given entity for self-authentication.
- Trusted certificate - contains a single public key certificate. It is called a trusted certificate because the keystore owner trusts that the public key in the certificate belongs to the identity identified by the subject (owner) of the certificate. This type of entry can be used to authenticate other parties.

Certificates of trusted entities are typically imported into a keystore as trusted certificates.

Keystore Entries and Aliases

Each entry in a keystore is identified by an *alias*. In the case of private keys and their associated certificate chains, these aliases distinguish among the different ways in which the entity may authenticate itself. For example, the entity may authenticate itself using different certificate authorities, or using different public key algorithms. An alias might be named after the role in which the keystore owner uses the associated key, or might identify the purpose of the key.

Keystore Passwords and Private Key Passwords

The private keys in a keystore are encrypted with a keystore password, which should be several words long.

You can also protect each private key with its individual password, which may or may not be the same as the keystore password.



If a password is lost, the associated keys cannot be recovered.

Creating a Keystore With a Username and Password

You can create a keystore that contains a username and password by editing data and build files and running an Ant script.

Procedure

1. Go to the `TIBCO_HOME/administrator/<version_number>/samples/` directory.
2. Open the `keystore_data.xml` data file and edit the following attributes of the `CredentialEntry` element:

Attribute	Description
alias	Alias identifying the keystore entry.
protectionParam	Password that protects the keystore entry.
username	Username.
secret	Password.

```
<?xml version="1.0" encoding="UTF-8"?>
<amxdata_base:Enterprise
  xmlns:amxdata="http://tibco.com/amxadministrator/command/line/types"
  xmlns:amxdata_base="http://tibco.com/amxadministrator/command/line/
types_base"
  xmlns:amxdata_reference="http://tibco.com/amxadministrator/command/line/
types_reference"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://tibco.com/amxadministrator/command/line/
types_base ../schemas/amxdata_base.xsd http://tibco.com/amxadministrator/
command/line/types ../schemas/amxdata.xsd">

  <AMXKeyStore xsi:type="amxdata:AMXKeyStore">
    <CredentialEntry alias="myDatabase"
protectionParam="databaseKeyAliasPassword" username="scott" secret="tiger" />
    <CredentialEntry alias="myLDAP"
protectionParam="ldapKeyAliasPassword"
username="cn=Manager,dc=example,dc=com" secret="password" />
  </AMXKeyStore>

</amxdata_base:Enterprise>

<target name="addCredential">
  <AMXKeyStoreTask
    dataFile="keystore_data.xml"
    adminKeyStoreLocation = "my_keystore.jceks"
    adminKeyStorePassword = "password"
    action="add"/>
</target>
```

```
>ant -f keystore_build.xml addCredential
Buildfile: C:\amx320\administrator\<version_number>\samples\keystore_build.xml
```

```
addCredential:
[AMXKeyStoreTask] INFO - Keystore file C:\amx320data\admin\amxadmin\samples
\my_keystore.jceks does not exist; creat
ing a new keystore file
[AMXKeyStoreTask] Adding entry for alias 'myDatabase'...
[AMXKeyStoreTask] Adding entry for alias 'myLDAP'...
[AMXKeyStoreTask] Saving to keystore file C:\amx320\administrator
\<version_number>\samples\my_keystore.jceks
```

```
BUILD SUCCESSFUL
Total time: 12 seconds
```

SSL Client Provider

The SSL Client Provider resource template maintains the credentials required by an SSL client.

General

Property	Required?	Editable?	Accepts SVars?	Description
Keystore Provider as Trust Store	Y	Y	N	The name of a Keystore Provider resource instance that maintains a keystore that confirms an identity.
Enable Access to Trust Store	N	N	N	<p>Enable access to a trust credential store.</p> <p>In order to establish SSL connections certain third-party systems, such as MySQL, require access to a keystore file location. In such situations Administrator provides a copy of the credentials in a keystore which are then written to disk and used by the third party as the SSL credential store.</p> <p>Default: Checked.</p>

Property	Required ?	Editable?	Accepts SVars?	Description
Enable Mutual Authentication	N	Y	N	<p>Indicate whether the client in the SSL connection will authenticate to the server. When checked, the identity fields are enabled.</p> <p>Default: Unchecked.</p>

Property	Required ?	Editable?	Accepts SVars?	Description
Identity Store Provider	Y	Y	N	Name of Keystore Provider resource that maintains a keystore used to assert an identity.

Property	Required ?	Editable?	Accepts SVars?	Description
Enable Access to Identity Provider	N	N	N	Enables access to an identity keystore. To establish SSL connections, certain third-party systems such as MySQL require access to a keystore file location. In such situations Administrator provides a copy of credentials in a keystore, which are then written to disk and used by the third party as the SSL credential store. To prevent Administrator from providing credentials, uncheck the checkbox. Default: Unchecked.
Key Alias Name	Y	Y	Y	Name of the alias used to access the identity. Default: None.
Key Alias Password	Y	Y	N	Password for the alias. (Administrator only) For superusers, passwords display encrypted. For non-superusers, the password doesn't display even if it was set when it was created. If you have permission to edit the password, you can specify a new value and save. If you edit other fields, the old value for the password field is retained. If you want to set an empty value as password, click the link Set Blank Password .

Advanced

Property	Required ?	Editable?	Accepts SVars?	Description
SSL Security Provider	N	Y	N	Optional. The SSL security provider.
SSL Protocol	N	Y	N	The SSL protocol to use in the SSL connection: <ul style="list-style-type: none"> • SSLv3 • TLSv1 • TLSv1.1 • TLSv1.2 Default: TLSv1.2.

Property	Required ?	Editable?	Accepts SVars?	Description
SSL Cipher Class	N	Y	N	<p>The number of bits in the key used to encrypt data:</p> <ul style="list-style-type: none"> • No Exportable Ciphers • At Least 128 Bit • More Than 128 Bit • At Least 256 Bit • FIPS Ciphers • All Ciphers • Explicit Ciphers <p>The greater the number of bits in the key (cipher strength), the more possible key combinations and the longer it would take to break the encryption.</p> <p>Default: At Least 128 Bit.</p>
Explicit Cipher List	N	Y	Y	<p>A list of ciphers. Enabled when SSL Cipher Class is set to Explicit Ciphers. Use the JSSE format for ciphers names.</p> <p>Default: None</p>
Verify Remote Hostname	N	N	N	<p>Indicate whether the name on the server's certificate must be verified against the server's hostname. If the server's hostname is different than the name on the certificate, the SSL connection will fail. The name on the certificate can be verified against another name by specifying Expected Remote Hostname. When checked, the Expected Remote Hostname field is enabled.</p> <p>Default: Unchecked.</p>
Expected Remote Hostname	N	Y	Y	<p>Optional. The expected name of the remote host.</p> <p>Default: None</p>

SMTP

An SMTP resource template represents a connection to an SMTP server. Used by component implementations to send and receive messages to and from an SMTP mail server.

General

Property	Required?	Editable?	Accepts SVars?	Description
Machine Name	Y	Y	Y	The name of the host that accepts incoming requests. Default: localhost.
Port	Y	Y	Y	The port number on which to listen for SMTP requests. Default: 25.
Timeout (ms)	N	Y	Y	The length of time to wait for a response from the server. The timeout must be greater than 0. A timeout of zero is interpreted as an infinite timeout. Default: 0.

Login Credentials

Property	Required?	Editable?	Accepts SVars?	Description
Login Credentials	Y	Y	N	Indicate how the credentials required to authenticate to a server are provided: <ul style="list-style-type: none"> • Identity Provider - Provide username and password credentials encapsulated in an identity provider resource. When selected, the Identity Provider field is activated. • Username + Password - Provide inline username and password credentials. When selected, the Username and Password fields are activated. Default: Identity Provider
Identity Provider	N	Y	N	Name of the Identity Provider resource used to authenticate the user.

Property	Required?	Editable?	Accepts SVars?	Description
Username	N	Y	N	Username used to authenticate connections to the server.
Password	N	Y	N	<p>User's password used to authenticate connections to the server.</p> <p>(Administrator only) For superusers, passwords display encrypted. For non-superusers, the password doesn't display even if it was set when it was created. If you have permission to edit the password, you can specify a new value and save. If you edit other fields, the old value for the password field is retained. If you want to set an empty value as password, click the link Set Blank Password.</p>

SSL

Property	Required ?	Editable ?	Accepts SVars?	Description
Enable SSL	Y	Y	N	<p>Enable SSL connections. When checked, the SSL properties display.</p> <p>Default: Unchecked.</p>
SSL Client Provider	Y	Y	N	The name of an SSL Client Provider resource.

Thread Pool

A thread pool is a queue of threads available to run a queue of tasks. Thread pools are used to improve performance when executing large numbers of asynchronous tasks by reducing per-task invocation overhead and provide a means of bounding and managing the resources consumed when executing a collection of tasks.

A thread pool is created with zero threads.

General

Property	Required?	Editable ?	Accepts SVars?	Description
Core Pool Size	N	Y	Y	<p>When a new task is submitted and fewer than Core Pool Size threads are running, a new thread is created to handle the request, even if other threads are idle. If there are greater than Core Pool Size but fewer than Max Pool Size threads running, a new thread is created only if no threads are idle. Must be greater than or equal to zero.</p> <p>Default: 2. Two threads are used to service one request: one for receiving the request and one for receiving the response.</p>
Max Pool Size	N	Y	Y	<p>The maximum number of threads in the pool. Must be greater than zero and greater than or equal to Core Pool Size.</p> <p>Default: 10.</p>
Keep Alive Time (s)	N	Y	Y	<p>The amount of time an idle thread remains in the pool before being reclaimed if the number of threads in pool is more than Core Pool Size.</p> <p>Default: 30 Seconds.</p>
Autostart Core Threads	N	N	Y	<p>Indicate that Core Pool Size threads should be created and started when the thread pool is created. Normally core threads are created and started only when new tasks arrive.</p> <p>Default: false.</p>
Thread Pool Name Prefix	N	Y	Y	<p>A string prepended to the name of each thread.</p> <p>Default: <pool-poolnumber-thread-threadnumber></p>
Priority	Y	N	Y	<p>The default priority of the threads in the pool.</p> <p>Default: 5.</p>

Property	Required?	Editable?	Accepts SVars?	Description
Rejection Policy	Y	N	N	<p>The policy applied when no thread is available to run a task:</p> <ul style="list-style-type: none"> • Abort - The task is aborted and an exception is thrown. • Blocking - The task is blocked until a thread from thread pool picks up this task. • Caller Runs - The task is run in the calling thread. <p>Default: Blocking.</p>
Daemon	N	N	Y	<p>Indicate whether the threads can be started as daemon or user.</p> <p>Default: Unchecked.</p>

Teneo

A Teneo resource is used by component implementations to access databases. Teneo is a model-relational mapping and runtime database persistence solution for the Eclipse Modeling Framework (EMF). Teneo integrates EMF with Hibernate.

General

Property	Required?	Editable?	Accepts SVars?	Description
Data Source	Y	Y	N	The name of a JDBC resource that represents the connection to the database.

Property	Required ?	Editable?	Accept s SVars?	Description
Schema Generation Type	N	N	N	<p>Indicate whether to create or validate the schema in the database when the session factory is created:</p> <ul style="list-style-type: none"> • Do Nothing - Indicate that only data is added, changed, and deleted. If the schema does not already exist, the application will experience errors when it runs. • Validate - Validate the schema. • Create - Create the schema every time the session factory is created, deleting old schema and data if it exists. • Create Drop - Same as Create, but drops the schema at the end of the session. • Update - Update the schema with the changes implied by the Java objects being mapped to the database. <p>Default: Do Nothing.</p>

Property	Required ?	Editable?	Accepts SVars?	Description
Dialect	Y	Y	Y	<p>The class name of a Hibernate dialect that enables Hibernate to generate SQL optimized for a particular relational database. The supported dialects are:</p> <ul style="list-style-type: none"> org.hibernate.dialect <ul style="list-style-type: none"> DB2390Dialect DB2400Dialect DB2Dialect FirebirdDialect FrontbaseDialect HSQLDialect InformixDialect IngresDialect InterbaseDialect MckoiDialect MySQLDialect MySQLInnoDBDialect MySQLMyISAMDialect Oracle9Dialect OracleDialect PointbaseDialect PostgreSQLDialect ProgressDialect SAPDBDialect SQLServerDialect SybaseAnywhereDialect SybaseDialect com.tibco.amf.sharedresource.runtime.core.hibernate.dialects <ul style="list-style-type: none"> DB2Dialect HSQLDialect MySQL5Dialect Oracle9Dialect Oracle10GDialect SQLServerDialect

Property	Required?	Editable?	Accepts SVars?	Description
				Default: com.tibco.amf.sharedresource.runtime.core. hibernate.dialects.HSQLDialect

Property	Required?	Editable?	Accepts SVars?	Description
Inheritance Mapping Type	N	N	N	<p>Indicate how class hierarchies are mapped to tables.</p> <ul style="list-style-type: none"> • SINGLE-TABLE The classes of one class hierarchy are all mapped to one table. • JOINED Each subclass has its own table. To retrieve an object from the database, the superclass and subclass tables are joined. This also applies to subclasses of subclasses. <p>Default: Single Table.</p>

Advanced

Property	Required?	Editable?	Accepts SVars?	Description
Enable SQL Logging	N	N	Y	<p>Permit data collection in the SQL Server transaction log file.</p> <p>Default: Unchecked.</p>
Batch Size	N	Y	Y	<p>Enables JDBC batch processing.</p> <p>Default: 5.</p>
Share Session Factory	N	Y	Y	<p>Indicate whether clients share the session factory or whether a new factory is created for each client.</p> <p>Default: Checked.</p>

Property	Required ?	Editable?	Accepts SVars?	Description
Properties	N	Y	N	<p>Hibernate configuration properties:</p> <ul style="list-style-type: none"> • Format SQL Enabled • Default Schema • Default Catalog • Max Fetch Depth • Default Batch Fetch Size • Use Order Updates • Use Order Inserts • Use Generate Statistics • Use Identifier Rollback • Use SQL Comments • Fetch Size • Batch Versioned Data • Batch Factory Class • Use Scrollable Resultset • Use Stream For Binary • Use Get Generated Keys • Connection Isolation • Use Auto Commit • Connection Release Mode • Cache Provider Class • Use Minimal Puts • Use Query Cache • Use Second Level Cache • Query Cache Factory • Cache Region Prefix • Use Structured Entries • Transaction Factory Class • JTA Transaction JNDI Name • Flush Before Completion • Auto Close Session • Query Factory Class • Query Substitutions • Use Reflection Optimizer

Properties

A property has a type, which may be either simple or complex. Implementations, components, composites, bindings, logging configurations and appenders, and resource templates can have properties. Implementation, component, and composite properties are defined in TIBCO Business Studio. Binding, logging configuration and logging appender, and resource template properties are defined by the TIBCO ActiveMatrix platform.

Properties can have explicit values or may be bound to substitution variables, which can be set at deployment time in various scopes. Depending on the object possessing the property, the property value can be bound at design time, deployment time, or both:

- At design time you can provide default values and indicate whether a composite or component property value must be set at deployment time.
- Some properties can be bound to substitution variables.

At design time, a composite property value can be set to a constant or bound to a substitution variable. Either type of binding can be overridden at administration time. However, only the properties of the root composite of an application or those on bindings associated with application level services and references can be overridden. If there are nested composites (component of type composite) then their property values cannot be changed by an Administrator.

A composite property is specific to an application. Often the same property may be defined in more than one application. For business reasons or ease of use an Administrator may want to define the value only once and have it be used by more than one composite property. This is achieved by binding the composite property to a substitution variable, which can be defined at the enterprise, host, environment, node, application, and application fragment levels.

The Owner column displays more contextual information about the owner of the property. Properties display a prefix indicating the context as follows:

- Application level properties display with the prefix [Application]
- Binding level properties display with the prefix [Service] or [Reference]
- Component level properties display with the prefix [Component]
- Properties at nested composites display with the prefix [Composite]
- Properties for certain policy sets such as Threading policy display with any of the preceding prefixes depending on where the policy set was added.

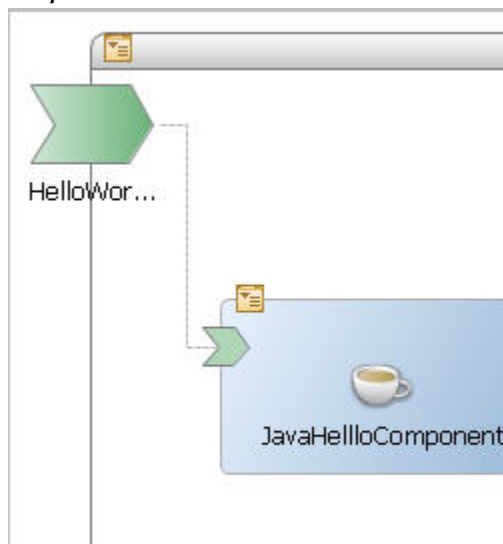
[Editable Properties](#) | [Non-editable Properties](#) | [Policy Set Properties](#)

Owner	Property Name	Property Type	Property Value
[Application]	test	string	test
	MEDIATION_VALIDATE_MESSAGE_DATA	boolean	false
[Service] Sample > SOAPService_Binding1	HttpInboundConnectionConfig	HttpConnector	httpConnector
[Reference] Sample1 > SOAPReference_Binding1	HttpOutboundConnectionConfig	HTTP Client	HttpClient_SampleSOAP
[Service] Sample > SOAPService_Binding1 > ThreadingPolicy_S	threadpool	Thread Pool	TH-Test

A component may be deployed to more than one node and you may want to have different values passed for a component property in every node. In such cases you would set the component property to a substitution variable, and set the substitution variable to different values on each node.

A property is represented by a property (🔑) icon. Each time you add a property to a composite or component another icon is added to the composite or component. If you click the icon, the property sheet for the property displays in the Properties view. The following figure shows a composite and component that both have a property.

Properties



Validation

A validation error displays when a component property is unbound. A validation warning displays when a composite property is unbound.

Creating a Property

You can create a property from the Palette, from the Properties view, or by right-clicking a composite or component.

You can only create a property for a composite or component.

Procedure

1. Choose an initial control and follow the appropriate procedure.

Control	Procedure
Palette	<ol style="list-style-type: none"> 1. In the Palette, click the property (📄) icon. 2. Click the composite banner or component. 3. Click the property icon. A property sheet displays with a property named property1.
Properties View	<ol style="list-style-type: none"> 1. Click a composite or component. 2. In the Properties view, click the Properties tab. 3. Click the plus (⊕) icon. A row is added to the Properties table with a property named property1.
Composite or component	<ol style="list-style-type: none"> 1. Right-click the composite or component and select Add > Property.

A property icon is added to the composite or component. If the property is added to a composite, a warning badge is added to the property indicating that the property does not have a value and is not referenced by a component property. If the property is added to a component, error badges are


added to the component and the property indicating that the component is out of sync with its implementation and the property does not have a value.

2. In the Name field or column, type a property name.
3. In the Type drop-down list or column, select a type.
4. To indicate that the property value is not required to be set at packaging time, uncheck the **Required** checkbox.

Promoting a Component Property

When you promote a component property, a property with the same name is created in the composite. You can promote an existing property from the properties icon.

Procedure


- Right-click a component property icon  and select **Tasks > Promote**.
If the composite does not have a property with the same name as the component property the composite property with the same name is created. If the property already exists, a property with the name with an integer appended is created. In both cases, property icon is added to the composite and the source of the component property is set to the composite property.

Setting the Value of a Binding Property

The process of setting the value of a binding property differs for simple properties and resource templates.

Procedure

- Follow the appropriate procedure according to the property type.






Type	Procedure
Simple	Type the value.
Resource Template	<p>Do one of the following:</p> <ul style="list-style-type: none"> • Type the name of a resource template. • Click . The Select XXX Resource Template dialog, where XXX is a resource template type, displays. <ul style="list-style-type: none"> – Click a resource template. – Click Create Resource Template and follow the procedure in Creating a Resource Template.

Setting the Value of a Composite Property

You can set the value of a composite property from the Properties tab or from the property icon. The procedure is different for simple properties and for resource templates.

Procedure

- Choose a control and follow the appropriate procedure.

Control	Procedure
Properties Tab	<ol style="list-style-type: none"> 1. Click the composite. 2. Display the Properties view. 3. Click the Properties tab. 4. Do one of the following: <ul style="list-style-type: none"> • Source the property to another object. <ol style="list-style-type: none"> 1. Click Source column in the property's row. 2. Click the ellipsis , click a substitution variable in the picker, and click OK. • Set the value according to the property type: <ul style="list-style-type: none"> – Simple - Type the value in the Value field. – Resource Template - Do one of the following: <ul style="list-style-type: none"> – Type the name of a resource template. – Click the ellipsis  icon. The Select XXX Resource Template dialog, where XXX is a resource template type, displays. <ul style="list-style-type: none"> – Click a resource template. – Click Create Resource Template and follow the procedure in Creating a Resource Template. <p>The property is filled in with the specified resource template.</p>
Property Sheet	<ol style="list-style-type: none"> 1. Click the property () icon on the composite. A property sheet opens in the Properties view. 2. Do one of the following: <ul style="list-style-type: none"> • Source the value to a substitution variable. <ol style="list-style-type: none"> 1. In the Substitution Variable field, click the -none- link or click the ellipsis  next to the field. The substitution variable picker displays. 2. Click a variable and click OK. The Substitution Variable field is replaced with the substitution variable name. The name is a link that when clicked opens the substitution variable editor. The Value field disappears. • Set the value according to the property type: <ul style="list-style-type: none"> – Simple - Type the value in the Value field. – Resource Template - Do one of the following: <ul style="list-style-type: none"> – Type the name of a resource template. – Click the ellipsis  icon. The Select XXX Resource Template dialog, where XXX is a resource template type, displays.




Control	Procedure
	<ul style="list-style-type: none"> – Click a resource template. – Click Create Resource Template and follow the procedure in Creating a Resource Template. <p>The property is filled in with the specified resource template.</p>




Setting the Value of a Component Property

You can set the value of a component property from the Properties tab or from the property icon. The procedure is different for simple properties and for resource templates.

Procedure

- Choose a control and follow the appropriate procedure.

Control	Procedure
Properties Tab	<ol style="list-style-type: none"> 1. Click the component. 2. Display the Properties view. 3. In the Properties view, click the Properties tab. 4. Do one of the following: <ul style="list-style-type: none"> • Source the property to an existing composite property. <ol style="list-style-type: none"> 1. In the property's row, click Source column. 2. Click the ellipsis . 3. Click a composite property in the picker. 4. Click OK. The component property value is bound to the composite property. • Source the property to a new composite property. <ol style="list-style-type: none"> 1. Right-click the property icon  on the component and select Promote. A property named <i>componentPropertyName</i> is added to the composite and the component property value is bound to the composite property. If the same property of the same type is already promoted by another component, the component property is bound to the existing property. • Set the value according to the property type: <ul style="list-style-type: none"> – Simple - Type the value in the Value field. – Resource Template - Do one of the following: <ul style="list-style-type: none"> – Type the name of a resource template. – Click the ellipsis () icon. The Select XXX Resource Template dialog, where XXX is a resource template type, displays. <ul style="list-style-type: none"> – Click a resource template.



Control	Procedure
	<ul style="list-style-type: none"> Click Create Resource Template and follow the procedure in Creating a Resource Template. <p>The property is filled in with the specified resource template.</p>
Property Sheet	<ol style="list-style-type: none"> Click the property () icon on the component. A property sheet opens in the Properties view. Do one of the following: <ul style="list-style-type: none"> Source the property to a composite property: <ol style="list-style-type: none"> In the Composite Property field, click the -none- link or click the ellipsis  next to the field. The Select Composite Property picker displays. If there are no properties, click Create Composite Property. The New Property dialog displays. <ol style="list-style-type: none"> In the Property Name field, type a name for the property and click OK. The property is added to the picker and is selected and the filter contains the created property name. Click OK. The component property value is bound to the composite property. The Composite Property field is replaced with the composite property name. The name is a link that opens the composite property editor when you click it. Set the value according to the property type: <ul style="list-style-type: none"> Simple - Type the value in the Value field. Resource Template - Do one of the following: <ul style="list-style-type: none"> Type the name of a resource template. Click the ellipsis () icon. The Select XXX Resource Template dialog, where XXX is a resource template type, displays. <ul style="list-style-type: none"> Click a resource template. Click Create Resource Template and follow the procedure in Creating a Resource Template. <p>The property is filled in with the specified resource template.</p>

Setting the Value of a Resource Template Property

Procedure

- Follow the procedure for type of property you want to modify.

Source	Procedure
Simple	Do one of the following:

Source	Procedure
	<ul style="list-style-type: none"> • Type the value. Some resource templates have properties that accept passwords. Passwords can be specified as clear or obfuscated text. • Set with a substitution variable: <ol style="list-style-type: none"> 1. Click the  icon. The Binding Editor displays listing the properties that can be bound to a substitution variable. <ol style="list-style-type: none"> a. Click a property. The row containing the property is highlighted. b. Click Pick.... The Select Substitution Variable picker displays listing the substitution variables whose type matches the selected property. c. Do one of the following according to whether the desired variable exists or not: <ul style="list-style-type: none"> • Click the variable and click OK. • Click Create new.... The Create XXX Substitution Variable dialog displays. <ol style="list-style-type: none"> 1. Click the substitution variable file in which to create the variable. 2. Click Next. 3. Click Add. 4. In the Name field, type the name of the variable. 5. In the Value field, type the value. 6. Click Finish. <p>A substitution variable string of the form <code>%%variableName%%</code> is added to the Binding column of the property.</p> 2. Click Close. The property value is set to the substitution variable and the property field is disabled.
Resource Template	<p>Do one of the following:</p> <ul style="list-style-type: none"> • Type the name of a resource template. • Click the ellipsis () icon. The Select XXX Resource Template dialog, where XXX is a resource template type, displays. <ul style="list-style-type: none"> – Click a resource template. – Click Create Resource Template and follow the procedure in Creating a Resource Template.

Creating an Obfuscated Password

Procedure

- Run the command: **ant -f TIBCO_HOME/administrator/<version_number>/samples/obfuscate_build.xml -Dpassword=*yourpassword***

```
C:\>ant -f C:\amx320\administrator\<version_number>\samples
\obfuscate_build.xml -Dpassword=mypw
Buildfile: C:\amx320\administrator\<version_number>\samples
\obfuscate_build.xml -Dpassword=mypw

encrypt:
[AMXObfuscateTask] INFO - Initializing JSSE's crypto provider class
com.sun.net
.ssl.internal.ssl.Provider in default mode
[AMXObfuscateTask] Obfuscated value:[#!EotHYBCR60hxS0l7VK0GqnyKeSAp0DVd]

BUILD SUCCESSFUL
Total time: 3 seconds
```

Properties Reference

You can set the name, type, source, and value of a property, and you can indicate whether the value is required.

Field	Description
Name	Name of the property. Default: Property1.
Type	Type of the property: String, Boolean, Double, Integer, or a resource template. Default: String.
Source	Source of the property value. For a component property, a literal or composite property. For a composite, resource template, or binding property, a literal or a substitution variable. Default: None.
Value	Property value. Default: None.
Required	The property value must be set before packaging the composite containing the property into a DAA. Default: Unchecked.

Substitution Variables

A *substitution variable* is a variable that you can reuse in resource, logging, and application configurations. Substitution variables enable late binding of property values to values set at administration time. For example, you can create an HTTP client resource template and bind its port property to a substitution variable that is set when the template is instantiated. The types of substitution variables are:

- String (default type)
- Integer
- Boolean
- Password

You can create substitution variables at design time and during administration. At design time, instead of explicitly setting property values, you can bind them to substitution variables. During administration, you set the substitution variables values to values supported by the resources available on the node on which the components and bindings are deployed.

A substitution variable is identified by a name. Names may not contain whitespace. When a property value is bound to a substitution variable, the property value is a string containing the substitution variable name surrounded by two pairs of percent signs.


Substitution variables are defined in substitution variable files. When you create an SOA project, a file named `SubstitutionVariables.substvar` is automatically added to the project in the `Resource Templates` special folder. By default, the file does not have any variables defined. You can also create additional substitution variables files.

Creating a Substitution Variable

You can create a substitution variable in the substitution variable editor or from a property binding.

Procedure

- Depending on the starting point, the steps to create a substitution variable differs.

Starting Point	Procedure
Substitution Variable Editor	<ol style="list-style-type: none"> 1. Click the plus () icon next to the substitution variable list. A new variable is added to the list. 2. Type a name in the Name field. 3. Select a type in the Type drop-down list. 4. Type a value in the Value field.
Property Binding	<ol style="list-style-type: none"> 1. Invoke substitution variable binding when setting a resource template or composite property value. The Select XXX Substitution Variable picker displays, where XXX is the type of the property. 2. Click Create new.... The Create XXX Substitution Variable dialog displays. <ol style="list-style-type: none"> a. Click the substitution variable file in which to create the variable. b. Click Next. c. Click Add.

Starting Point	Procedure
	<ol style="list-style-type: none"> d. In the Name field, type the name of the variable. e. In the Value field, type the value. f. Click Finish.

Creating a Substitution Variable File

You can create a substitution variable file from the project explorer or by invoking a substitution variable binding when you set a resource template or composite property value.

Procedure

- Depending on the starting point, the steps to create a substitution variable file differs.

Starting Point	Procedure
Project Explorer	<ol style="list-style-type: none"> 1. Select File > New > Other > TIBCO Shared Resources > Substitution Variable Resource or In the Project Explorer, right-click an <i>SOAPProject/Resource Templates</i> folder to contain the file and select New > Substitution Variable Resource. The Create SVar File dialog displays. 2. Select or enter a folder to contain the file. 3. In the File name field, type a name for the file. 4. Click Finish.
Property Binding	<ol style="list-style-type: none"> 1. Invoke substitution variable binding when setting a resource template or composite property value. 2. Click Create new... 3. Click Create SVar File. 4. Select or enter a folder to contain the file. 5. In the File name field, type a name for the file. 6. Click Finish.

Finding References to a Substitution Variable

When you have to find references to a substitution variable, you can do so from the substitution variables file.

Procedure

1. Open a substitution variables file.
2. Click a substitution variable.
3. In the right pane, click the **Find References** link to the right of the variable name.
A list of objects that reference the variable display in the Search view.

Renaming a Substitution Variable

You can rename a substitution variable from the substitution variable file. You can optionally update the selected referencing objects as part of the process.

Procedure

1. Open a substitution variable file.
2. Click a substitution variable.
3. In the Name field, type a new name.
A dialog displays with actions related to references to the variable.
4. Press **Enter**.
The Rename Substitution Variable dialog displays.
5. To update all references to the resource, leave the **Update references** checkbox checked. Otherwise, uncheck the checkbox.
6. Click the **Preview** button.
A preview pane displays with a list the effect of the change. If the Update references checkbox is checked, the effects on objects that reference the renamed resource are listed.
7. Uncheck any changes you don't want and click **OK**.
The substitution variable is renamed. If the Update references checkbox was checked, the selected referencing objects are updated.

Custom Features

A *feature* is a software package that contains plug-ins, which in turn contain component implementations and libraries. A feature is identified by an ID, a multi-part version, and its dependencies on other features. In TIBCO Business Studio, a feature is described by a *custom feature file*.

Custom Feature Files

A TIBCO Business Studio feature is described by a *custom feature file*, which stores feature properties such as its ID, name, and version, its plug-ins (bundles), and the features on which it depends. You edit custom feature files in a custom feature editor.

Creating a Custom Feature

A custom feature is created automatically when you create certain types of component implementations. You can also manually create a custom feature to contain a shared library.

Procedure

1. Choose an initial control and follow the relevant procedure.

Control	Procedure
Menu	1. Select File > New > TIBCO SOA Resources > Custom Feature .
Canvas	1. Right-click a component whose feature is not configured, and select Quick Fixes > Create Custom Feature .
Problems View	<ol style="list-style-type: none"> 1. In the Problems view, right-click an error of the form There is no custom feature defined for the XXX component "<i>ComponentName</i>" and select Quick Fix. 2. In the Quick Fix dialog, click Create Custom Feature. 3. Click Finish.

2. In the dialog, select a folder to contain the feature.
3. Accept the default file name or type a new name. Click **Next >**.
4. Accept or edit the feature details according to [Custom Feature Reference](#).
5. Click **Next**.
6. In the left pane, click one or more plug-ins to package in the feature and click **Add**. The plug-ins move to the right pane.
7. Click **Finish**.

Finding References to a Custom Feature

You can find references to a custom feature from the feature editor.

Prerequisites

Open the custom feature editor.

- In the Project Explorer view, double-click the custom feature file.

- Right-click the file and select **Open With > Custom Feature Editor**

Procedure

1. Under the Name field, click the ***n* reference** link, where *n* is the number of references to the custom feature.
2. Examine the list of objects that reference the custom feature displays in the Search view.

Renaming a Custom Feature

You can rename the name or the reference of custom feature in a resource editor. The rename action optionally updates all references to the custom feature.

Prerequisites

Open the custom feature editor.

- In the Project Explorer view, double-click the custom feature file.
- Right-click the file and select **Open With > Custom Feature Editor**

Procedure

1. Choose a control and follow the appropriate procedure.

Control	Procedure
Name	<ol style="list-style-type: none"> 1. In the Name field, type a new name. 2. Press Enter.
Rename reference	<ol style="list-style-type: none"> 1. Click the Rename reference link. 2. In the New name field, type a new name.

2. Click **Preview**.
A preview pane displays with a list of the effects of the change. If the Update references checkbox is checked, the effects on objects that reference the renamed resource are listed.
3. Uncheck any changes you don't want and click **OK**.
The custom feature is renamed. If the Update references checkbox was checked, the selected referencing objects are updated.

Configuring Plug-ins

You can configure custom feature plug-ins in the custom feature editor.

Prerequisites



Open the custom feature editor.

- In the Project Explorer view, double-click the custom feature file.
- Right-click the file and select **Open With > Custom Feature Editor**

Procedure

1. Click the **Plug-ins** tab.

2. Choose an operation and follow the appropriate procedure.

Operation	Procedure
Add	<ol style="list-style-type: none"> 1. Click . 2. Click a plug-in. 3. In the Version field, specify a version.
Delete	<ol style="list-style-type: none"> 1. Click a plug-in. 2. Click .

3. Click **OK**.

Configuring Feature Dependencies

You can configure feature dependencies in the custom feature editor. Several operations are supported.




Prerequisites


Open the custom feature editor.

- In the Project Explorer view, double-click the custom feature file.
- Right-click the file and select **Open With > Custom Feature Editor**

Procedure

1. Click the **Dependencies** tab.
2. Choose an operation and follow the appropriate procedure.

Operation	Procedure
Add from workspace	<ol style="list-style-type: none"> 1. Click . 2. Click a feature. 3. Click OK.
Add from target platform	<ol style="list-style-type: none"> 1. Click . 2. Click a feature. 3. Click OK.
Add manually	<ol style="list-style-type: none"> 1. Click . 2. Specify a feature ID. 3. Click OK. The feature is added with version range [1.0.0, 2.0.0).
Delete	<ol style="list-style-type: none"> 1. Click a feature.



Operation	Procedure
	2. Click  .

Configuring Excluded Custom Feature Dependencies

When you package a composite or custom feature in a DAA, you can exclude custom feature dependencies so custom features do not appear.

Procedure

1. Select **Windows > Preferences**.
2. Select **TIBCO SOA Platform > Excluded Custom Feature Dependencies**.
3. Choose an operation and follow the appropriate procedure.

Operation	Procedure
Add	<ol style="list-style-type: none"> 1. Click . A new row is added to the table with an ellipsis (...). 2. Click ... and type the name of the feature.
Delete	<ol style="list-style-type: none"> 1. Click a row containing a feature. 2. Click .

4. Click **OK**.

Custom Feature Reference

Overview Tab

Field	Description
ID	<p>The ID of the custom feature.</p> <p>Default:</p> <ul style="list-style-type: none"> • Auto-generated - <i>CompositeName</i>.customfeature.id. • Wizard - NewCustomFeature.feature
Name	<p>Translatable display name of the custom feature.</p> <p>Default:</p> <ul style="list-style-type: none"> • Auto-generated - <i>CompositeName</i>.customfeature.id. • Wizard - NewCustomFeature feature
Version	<p>The feature version number.</p> <p>Default: 1.0.0.qualifier.</p>

Field	Description
Provider	Optional. Translatable name of the custom feature provider. Default: <i>username</i> .
Description	Optional description of the custom feature.

Plug-ins Tab

The plug-ins (bundles) packaged in the custom feature.

Field	Description
Version	The version of the plug-in packaged in the custom feature.
Unpack	Indicate whether the plug-in should be unpacked when deployed to a node. A plug-in needs to be unpacked if it contains library JARs that an application must access by file name. Default: Unchecked.

Dependencies Tab

The features on which the custom feature depends. Before the feature is installed, Administrator ensures that the feature's dependencies can be satisfied on the node on which the feature will be installed.

Feature Dependencies

Field	Description
Compute Dependent Features	Indicate that when you package the custom feature in a DAA, TIBCO Business Studio will compute and display the features on which the feature depends. To preview the computed dependent features in the Computed Dependency group in the Features list, click the Preview link. Default: Checked.
Features	A list of the features on which the custom feature depends: <ul style="list-style-type: none"> Manually Added Dependency - The dependent features manually added to the feature. Computed Dependency - The dependent features computed by TIBCO Business Studio.

Dependency Properties

Field	Description
Version	The version of the dependent feature.

Field	Description
Match Rule	<p>A drop-down list of match rules that determine whether the version of an available feature matches the version of the required feature. If the version attribute is not specified, the match rule is ignored. The match rule can take the following values:</p> <ul style="list-style-type: none">• Perfect - The version must match exactly the specified version.• Equivalent - The major and minor version numbers must match the specified version.• Compatible - The major version number must match the specified version.• Greater Or Equal - The version numbers must be at least the version specified or higher.

Shared Libraries

TIBCO ActiveMatrix products support the ability to package and deploy libraries that can be shared between multiple applications.

There are several scenarios for shared libraries:

- Data binding libraries generated by TIBCO Business Studio
- Libraries containing a third-party JAR
- Libraries that you develop in TIBCO Business Studio

A shared library must be contained in a Java plug-in project.

Bundles and Plug-in Projects

A *bundle* is an OSGi mechanism for grouping Java classes into a modular, sharable unit. In TIBCO Business Studio, a plug-in project implements a bundle.

Plug-in properties, including the packages it exports and the objects on which it depends, are described in the plug-in manifest. The manifest file is located in *plug-inFolderName*META-INF/MANIFEST.MF. The default editor for the file is a manifest editor which displays OSGi headers in property sheets and in the MANIFEST.MF source view. [Plug-in Project Best Practices](#) summarizes the best practices you should follow when configuring plug-ins.

Plug-in Project Best Practices

Property	Manifest Editor UI	OSGi Header in Source View	Best Practice
Unique Identifier	Overview > ID	Bundle-SymbolicName	Give the plug-in a symbolic name that follows Java package name conventions. That is, <i>com.companyName.plug-inName</i> .
Version	Overview > Version	Bundle-Version	Follow the recommendations in Versions .
Display Name	Overview > Name	Bundle-Name	Give the plug-in an appropriate, descriptive display name.

Property	Manifest Editor UI	OSGi Header in Source View	Best Practice
Dependencies	Dependencies > Imported Packages Required Plug-ins	Import-Package Require-Bundle	<ul style="list-style-type: none"> Express dependencies based on the contents of the plug-in: <ul style="list-style-type: none"> For plug-ins that you create or if you want tight control of the dependency, specify the dependency as a required bundle. You can (but are not required to) indicate a perfect match to a specific plug-in and build. For example, when TIBCO Business Studio generates a component implementation, the component's dependency on the plug-in containing the component implementation is expressed as [1.0.0.qualifier,1.0.0.qualifier]. For third-party plug-ins, specify the dependency as an imported package. To allow packages to be upgraded without requiring plug-ins dependent on those packages to be upgraded, specify package dependency ranges of the form [x.y.z,x+1.0.0). That is, up to, but not including the next major version. For example, [2.3.0, 3.0.0). Minimize or eliminate optional imports.
Exported Packages	Runtime > Exported Packages	Export-Package	<ul style="list-style-type: none"> Export only those packages imported by other plug-ins. Put classes that are not exported in packages that are not exported. Specify versions of all exported packages. Import all exported packages, with a range floor of the exported package version, and a range ceiling of the next major version exclusive, per the range definition above. If the classes in the exported packages use other classes (for example, they extend classes from another package or the classes appear in method signatures) add the uses directive to the export package definition.

Creating a Library Plug-in Project

You can create a library plug-in from a third-party JAR file or a class source. After you create the plug-in, you can create a custom feature for it or export it as a DAA.

Procedure

1. Follow the appropriate procedure for the plug-in project contents.

Project Contents	Procedure
Third-Party JAR	<ol style="list-style-type: none"> 1. Select File > New > Project > Plug-in Development > Plug-in from existing JAR archives. 2. Click Next. 3. Click Add External. 4. Navigate to the folder containing the JAR, click the JAR, and click Open. 5. Click Next >. 6. In the Project name field, type a project name. 7. In the Plug-in ID field specify a plug-in ID. 8. In the Plug-in Version field accept the default or specify a correctly formatted version. 9. In the Target Platform area, click the an OSGi framework radio button and select Equinox from the drop-down list. 10. Uncheck the Unzip the JAR archives into the project checkbox.
Class Source	<ol style="list-style-type: none"> 1. Select File > New > Project... > Plug-in Development > Plug-in Project. 2. In the Project name field, type a project name. 3. In the Target Platform area, click the an OSGi framework radio button and select Equinox from the drop-down list. 4. Click Next. 5. In the Version field, accept the default or specify a correctly formatted version. 6. In the Options area, uncheck the Generate an activator checkbox. 7. Click Next. 8. Uncheck the Create a plug-in using one of the templates checkbox.

2. Click **Finish**.
The Open Associated Perspective dialog displays.
3. Click **No**.
A plug-in project folder is added to the Project Explorer. The plug-in manifest opens in a manifest editor.
4. If you are packaging class source, import the classes into the project's **src** folder.
 - a) Right-click the **src** folder.
 - b) Click **Import**.
 - c) Click **File System** and click **Next**.
 - d) Click the **Browse** button and navigate to a folder containing the source files.
 - e) Check the checkboxes next to the files to import.

- f) Click **Finish**.
5. Export the packages.
 - a) In the Editor, click the **Runtime** tab.
 - b) Click **Add...** to the right of the Exported Packages table.
 - c) Select the packages containing the classes to be exported by the library.
 - d) Click **OK**.
 - e) Click the **Properties** button.
 - f) In the Version field, specify a package version.
 - g) Click **OK**.
6. If you are packaging an Oracle database driver library:
 - a) Click the **Dependencies** tab.
 - b) In the Imported Packages area, click **Add**.
 - c) In the Package Selection dialog, click **org.ietf.jgss**.
 - d) Click **OK**.
7. Click **Save**.

What to do next

- [Create a custom feature containing the plug-in.](#)
- [Export the custom feature as a DAA.](#)

Distributed Application Archives

You can package a shared library as a distributed application archive. A *distributed application archive* (DAA) is a package that contains TIBCO ActiveMatrix applications and libraries.

A DAA contains zero or one application template, zero or more features, and zero or more resources. When you upload a DAA, Administrator extracts the contents of the DAA and stores them in the Administrator staging area. The original DAA file is not stored, so the only way to delete a DAA is to delete each entity contained in the DAA.

By default, DAAs are stored in the `Deployment Artifacts` special folder in an SOA project.

Packaging a Composite in a DAA

You can package a composite from the canvas, the project explorer, or the distribution editor. A wizard steps you through the packaging steps.

Procedure

1. Choose a starting point and follow the appropriate procedure:

Starting Point	Procedure
Canvas	1. Right-click the canvas and select Create DAA...
Project Explorer	1. Right-click the composite file and select Create DAA...
Distribution Editor	1. Click the Create DAA... button in the top-right of the editor.

2. Accept the default folder to contain the archive, or type or select a new folder.
3. Accept the default file name or type a new name. Click **Next**.
4. Select one of the following options and click **Next**.
 - Do not use a distribution file.
 - Create a new distribution file. If you create a distribution file, the wizard invokes the [New Distribution](#) wizard.
5. Uncheck the checkboxes next to resource templates you do not want to package in the archive and click **Next**.
The Features screen displays the composite's dependencies on plug-ins packaged in custom features. If the code is present in the workspace, by default the wizard packages the custom features together with the application.
6. Choose the features to package in the archive and click **Next >**. By default, the features containing component implementations and features required by the component implementations are included. Features that require a different version of the platform on the node are not displayed.
 - Remove a feature - click the feature and click **Remove**.
 - Add a feature - click **Add**, click a feature, and click **OK**.

You can select additional custom features to package in the DAA. Added custom features might depend on other product features.

7. Choose the DAA specification options and whether to generate CLI scripts:
 - a) Choose whether to save the [DAA specification](#) file.

- b) To change the qualifier replacement string, check the Qualifier Replacement checkbox and edit the [Qualifier Replacement](#) field.
 - c) To launch a wizard to create an [Administrator Command-Line Interface](#) after the composite is packaged, check the **Launch Administrator CLI Script Generation Wizard** checkbox.
 - d) To save the specified DAA configuration as a [TIBCO Business Studio Command-Line Interface](#) script, check the **Save Configuration as CLI Script** checkbox.
8. Click **Finish**.

Result

A DAA file is created in the specified folder. If you chose to launch the Administrator CLI script wizard, the wizard displays. If you chose to save the specified DAA configuration as a TIBCO Business Studio command-line script, a build file named *DAAName.build.xml* and a properties file named *DAABuild.properties* is created in the specified folder. The build file contains targets to import all of the required projects, generate a DAA, and create an Administrator CLI script to deploy the DAA. The properties file contains the paths to the component and component implementation projects and sets the deployment environment and node to DevEnvironment and DevNode.

Packaging a Custom Feature in a DAA

You can package a custom feature by right-clicking the feature. As part of the process, you can choose to generate CLI scripts.

Procedure

1. Do one of the following:
 - Right-click the custom feature and select **Create DAA...**
 - Right-click the custom feature and select **Export > Custom Feature As DAA**.
2. Type a folder name or select a folder from the folder tree.
3. Type a name of the format *archiveName.daa* and click **Next**.
4. Choose whether to generate CLI scripts:
 - a) To launch a wizard to create an [Administrator Command-Line Interface](#) after the composite is packaged, check the **Launch Administrator CLI Script Generation Wizard** checkbox.
 - b) To save the specified DAA configuration as a [TIBCO Business Studio Command-Line Interface](#) script, check the **Save Configuration as CLI Script** checkbox.
5. Click **Finish**.

Result

A DAA file is created in the specified folder. If you chose to launch the Administrator CLI script wizard, the wizard displays. If you chose to save the specified DAA configuration as a TIBCO Business Studio command-line script, a build file named *DAAName.build.xml* and a properties file named *DAABuild.properties* is created in the specified folder. The build file contains targets to import all of the required projects, generate a DAA, and create an Administrator CLI script to deploy the DAA. The properties file contains the paths to the component and component implementation projects and sets the deployment environment and node to DevEnvironment and DevNode.

Regenerating a Distributed Application Archive Containing a Composite

You can regenerate a DAA by right-clicking the specification file.

Prerequisites

Save a [DAA specification](#) file when you package the composite in the DAA.

- Right-click a DAA specification file and select **Regenerate DAA**.

Generating and Verifying Derived DAAs

To generate and verify derived DAAs, you perform two tasks. First you configure TIBCO Business Studio to mark the DAA as derived, and then you verify that the DAA is marked as derived. If a DAA is marked as not derived, Subversion will identify a modified DAA as a candidate for check-in. To prevent Subversion from identifying DAAs as candidates for check-in, you can configure TIBCO Business Studio to mark generated DAAs as being derived.

Procedure

1. Configure TIBCO Business Studio to mark DAAs as derived
 - a) Select **Window > Preferences > TIBCO SOA Platform > Runtime Artifacts**.
 - b) Check the **Generate derived DAAs** checkbox.
DAAs generated by TIBCO Business Studio are marked as derived.
2. Verify that the DAA is marked as derived.
 - a) Right-click the DAA in the Project Explorer and select **Properties**.
 - b) In the dialog, verify that the checkbox labeled Derived is checked.

Distributed Application Archive Specifications

A distributed application archive (DAA) specification persists the configuration choices specified when running the DAA wizard. You can save the DAA specification file if you want to regenerate the DAA without rerunning the create DAA wizard. The specification includes only the custom features and shared resources that were selected for packaging when the specification was generated. If an updated application template requires additional custom features or shared resources, those resources will not be packaged into the DAA when the specification file is used to repackaging it.

Qualifier Replacement

If the *qualifier* component of a version is set to "qualifier" when you create a DAA, TIBCO Business Studio replaces the string "qualifier" with a generated qualifier that defaults to a timestamp. You can replace the qualifier if you use the proper format.

Qualifier strings must sort in increasing order. If you upload a DAA or deploy an application containing an object whose version is older than one currently available in the Administrator software repository, the upload or deployment operation will fail.

Qualifier Replacement Templates

A qualifier string can be a string such as "001" or be generated according to a qualifier replacement template. A *qualifier replacement template* specifies the formatting of the qualifier string. The default qualifier replacement template is a timestamp of format 'v'yyyy-MM-DD-HHmm, which yields a string such as v2011-07-27-1010. The basic qualifier replacement template formatting options are:

- yyyy - year
- MM - month
- dd - day of the month
- HH - hour of the day (0-23)
- mm - minute
- ss - second

The full set of timestamp formatting options are specified by the [java.text.SimpleDateFormat](#) class.

To append text after the timestamp, specify a dash and the text enclosed in single quotes. For example, 'v'yyyy-MM-DD-HHmm-'dev' yields the string v2011-07-27-1010-dev. To append a username or

hostname, specify `${user}` and `${host}` respectively. For example, `v'yyyy-MM-DD-HH:mm-'dev'-${user}` yields `v2011-07-27-1010-username`. `${host}` is substituted by your hostname or localhost if it cannot be determined.

Distributions

A *distribution* defines a mapping of one or more composite elements to one or more logical nodes. A *logical node* is a heterogeneous group of composite elements that must be deployed to the same physical node.

Distributions allow you to specify constraints on how composite elements are deployed. For example, you can specify that all components are distributed to one node and all bindings to another node. Distributions are optional. A distribution is created and edited in the Distribution Editor and saved to a distribution file.


Once a distribution file has been created, you can generate a distributed application archive (DAA) from it.

Creating a Distribution

You can create a distribution from the menubar, the project explorer, or the canvas. After you have completed the wizard, a distribution file is generated.


Procedure

1. Do one of the following:

Starting Point	Procedure
Menubar	<ol style="list-style-type: none"> 1. Select File > New > TIBCO SOA Resources > Distribution. 2. In the Enter or select the parent folder field, accept the default folder to contain the distribution or type or select a new folder. 3. In the File name field, accept the default file name or type a new name. 4. Click Next. 5. In the Composite field, click the . <ul style="list-style-type: none"> • Click a composite. • Click Create Composite. <ol style="list-style-type: none"> 1. In the Enter or select the parent folder field, accept the default folder to contain the composite or type or select a new folder. 2. In the File name field, accept the default file name or type a new name. 3. Click Finish. 6. Click OK.
Project Explorer	<ol style="list-style-type: none"> 1. Right-click a composite file and create a DAA. 2. In the Select Distribution screen of the Create Deployment Archive wizard click the Create a new distribution file radio button. 3. In the Enter or select the parent folder field, accept the default folder to contain the distribution or type or select a new folder. 4. In the File name field, accept the default file name or type a new name. 5. Click Next.

Starting Point	Procedure
Canvas	<ol style="list-style-type: none"> 1. Right-click a composite and select Create Distribution.... 2. In the Enter or select the parent folder field, accept the default folder to contain the distribution or type or select a new folder. 3. In the File name field, accept the default file name or type a new name. 4. Click Next.

The Logical Environment screen displays.

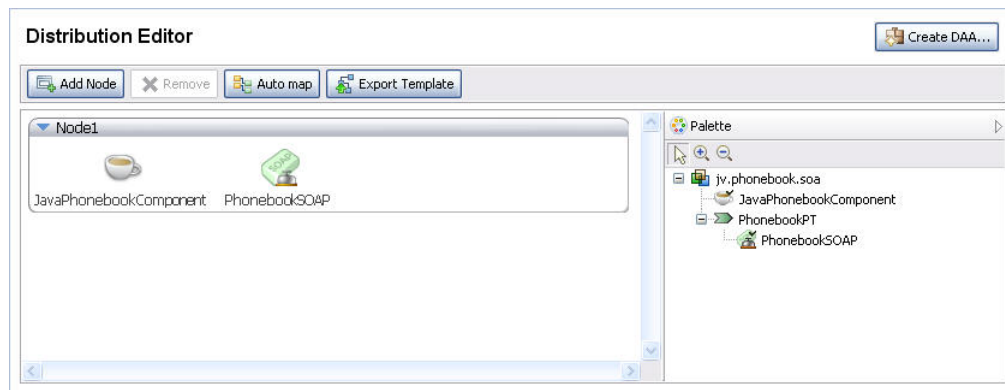
2. In the Creation Pattern drop-down list, select the pattern that determines how components and bindings are distributed to logical nodes:
 - **Single Node** - Components and bindings are distributed to a single logical node.
 - **Use Template** - Components and bindings are distributed according to a logical environment specified in a distribution template.
 1. In the Template field, click .
 2. Click a logical environment.
 3. Click **OK**.
 - **No Logical Environment** - The components and bindings are not mapped to logical nodes.
3. Click **Next**
The Distribution Strategy screen displays.
4. In the Distribution Strategy drop-down list, select the strategy for mapping components and bindings to logical nodes:
 - **All Compatible Nodes** - Bindings and components are replicated on all compatible logical nodes.
 - **First Compatible Node** - Components and bindings are distributed to the first logical node that satisfies their requirements.
 - **No Mapping** - Components and bindings are not mapped to logical nodes.
5. Uncheck the **Map Bindings** checkbox to remove the mapping of bindings to the logical nodes.
6. Click **Finish**.
A distribution file named *FileName.dist* is generated.

Editing a Distribution

You can add nodes to a distribution or specify the strategy for distribution components and bindings to logical nodes in the Distribution Editor.

Procedure

1. In the Project Explorer, double-click a distribution file.
The file is opened in the Distribution Editor.



2. Choose and operation and follow the appropriate procedure.

Operation	Procedure
Edit logical environment	<ul style="list-style-type: none"> Click Add Node to add a node to the logical environment. Click a node and click Remove to remove a node.
Edit mapping of components and bindings to logical nodes	<ul style="list-style-type: none"> Click Automap to specify the strategy for distributing components and bindings to logical nodes. Click a component or binding the Palette, and holding the left mouse button down, drag to a logical node, and release the button.

Creating a Distribution Template

You can create a distribution template in the Distribution Editor.

Procedure

- In the Project Explorer, double-click a distribution file.
The distribution file opens in the Distribution Editor.
- Optionally add one or more nodes to the canvas.
- Click **Export Template**.
The Export Template Configuration dialog displays.
- In the Enter or select the parent folder field, accept the default parent folder or type or select a new folder.
- In the File name field, accept the default name or type a new one.
- Click **Next**.
- Choose a template configuration.
 - Use nodes in canvas - exports the logical environment displayed in the canvas.
 - Export template in distribution - exports the logical environment chosen in the Creation Pattern field of the Logical Environment screen of the Distribution wizard.
- Click **Finish**.
A logical environment template with the extension .logical is created in the folder specified above.

Applications

TIBCO ActiveMatrix SOA applications assume different forms in different phases of the application life cycle. The cycle consists of a design phase, an administration phase, and a runtime phase.

In the design phase a TIBCO ActiveMatrix SOA application consists of one or more composites. Each application has a root composite. A composite contains components, services, references, and properties. The components, services, and references depend on custom features and resources. Services, references, and properties promoted to the root composite comprise the public interface of the application.

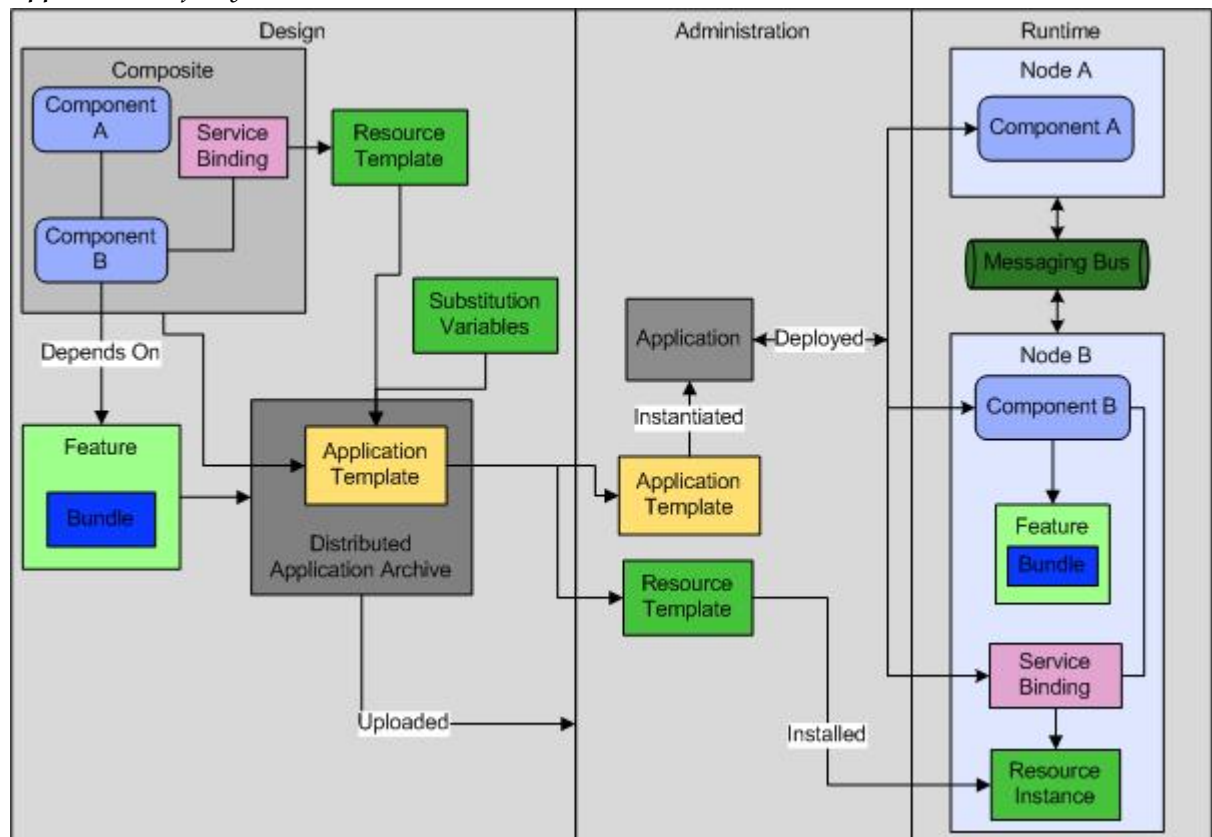
The output of the design phase is a distributed application archive (DAA). A DAA contains features and an *application template*, which consists of the root composite and a set of related configuration files:

- Nested composites
- Resource templates
- WSDL files and
- Substitution variable files

In the administration phase, you create an application by instantiating an *application template*. The product installer adds product application templates to Administrator. When you upload a DAA file to Administrator, Administrator extracts the application template and (optionally) the features and resource templates.

The following figure illustrates the application artifacts across the application life cycle.

Application Life Cycle



The following actions can be performed using the Administrator:

- Configure the application by setting properties, logging configurations, and substitution variables

- Add bindings to services
- Promote services and references to the environment
- Wire services and references to services and references in other applications or environments
- Specify a distribution of the application to the runtime infrastructure
- Explicitly distribute application fragments-components and bindings-to one or more nodes
- Specify that an application should be distributed to the same nodes as another application

When you deploy the application, Administrator applies the distribution and the configuration. Features are automatically distributed with components, but resource instances required by the application must be manually installed into nodes before deployment. Life cycle operations on the application are translated into life cycle operations on the application fragments. You can also start and stop application fragments.

Creating, Deploying, and Starting an Application

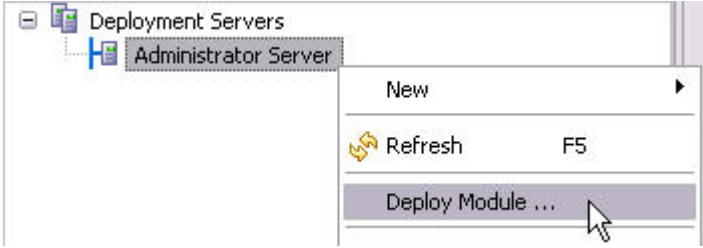
With a DAA available, you can create, deploy, and start the corresponding application from the Deployment Server or from the Project Explorer.

Prerequisites

Create and connect to an Administrator Server.

Procedure

1. Choose a view and follow the appropriate procedure:

View	Procedure
Deployment Server	<ol style="list-style-type: none"> 1. Right-click a deployment server and select Deploy Module.  2. Choose the DAA containing the application template that will be used to create the application. Click Workspace or Filesystem and click the DAA.
Project Explorer	<p>Do one of the following:</p> <ul style="list-style-type: none"> • 1. Click a DAA, drag to a deployment server in the Deployment Server view, and release the mouse button. • 2. Click Next. • 1. Right-click a DAA and select Generate Administrator CLI Script... • 2. Complete Script Configuration selecting Start Application in the Default Action drop-down list and check the Execute generated script checkbox.

View	Procedure
	3. Complete Administrator Connection Configuration selecting the deployment server you created and connected to in the task prerequisite.

2. Complete the wizard screens and click **Next** to advance to each screen.
3. Click **Finish**.

Result

An Administrator CLI script that contains actions to upload a DAA, and configure, deploy, and start the application is generated and executed.

Stopping, Undeploying, and Deleting an Application

You can stop, undeploy, and delete an application from the Administrator Explorer view. After you perform the setup, Administrator generates and executes a CLI script that performs the corresponding actions.

Prerequisites

Create and connect to an Administrator Server.

Procedure

1. In the Administrator Explorer view, expand the *ServerName > EnvironmentName > Applications* node.
2. Right-click an application and select **Delete Application**
3. Click **OK**.
An Administrator CLI script that contains containing actions to stop, undeploy, and delete the application is generated and executed. The script sets a flag so that it will proceed with all delete actions even if it references objects that do not exist.

Debugging Deployed Applications

TIBCO Business Studio provides support for debugging deployed applications.

Debugging a deployed application consists of two tasks.

- Create a debug configuration in which you specify one or more running applications and remote debuggers and launch the debug configuration.
- Attach a debugger to a running application, which automatically creates and launches the debug configuration.

Before debugging a remotely deployed application, you must enable debuggers on the nodes on which the application is deployed. For information on how to enable a debugger, see *Administration*.

Enabling a Java debugger will increase the time it takes to receive responses to requests sent to applications running on the node.



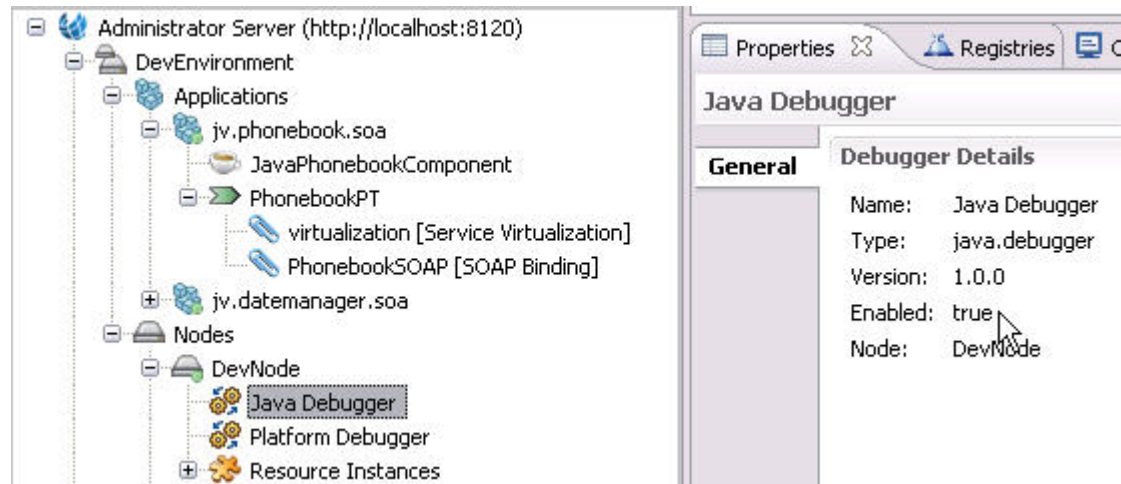
Do not enable debuggers in production systems. A rogue process could attach to a debugger and halt the node.

Checking Remote Debugger Status

You can check remote debugger status in the Administrator Explorer.

Procedure

1. In the Administrator Explorer view, expand the nodes of the Administrator server that manages the application.
2. Click a debugger under a node.
3. In the General tab of the Properties view, confirm that the Enabled field is true.



Creating a Debug Configuration

Procedure

1. Select **Run > Debug Configurations**.
The Debug Configurations dialog displays.
2. Double-click **Remote Composite Application**.
A new debug configuration named New_configuration is created and opened in the configuration editor.
3. In the Name field, type a configuration name.
4. In the Remote Applications tab click **Add...**



The available applications are imported from the Administrator server when you connect to the server. If you create applications after you connect, they will not appear in the dialog. To load the current applications, click **Refresh** before you click **Add**.

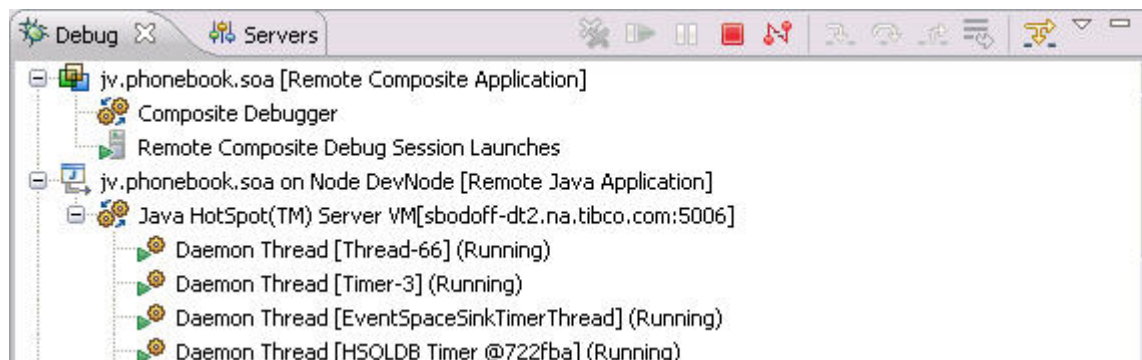
- a) Click the server that manages the application to be debugged and click **Next >**.
 - b) Click one or more applications and click **Finish**.
5. Click the **Common** tab.
 6. In the Display in favorites menu, check the **Debug** checkbox.
 7. Click **Apply** and **Close**.
The configuration is saved and added to the Debug menu options.

Launching a Debug Configuration

After the debugger has created a debug configuration, you can use it to debug the running application.

Procedure

1. Select **Run > Debug Configurations...** .
The Debug Configurations dialog displays.
2. Double-click a Remote Composite Application configuration.
The configuration opens in the configuration editor.
3. In the configuration editor, click **Debug**.
The debugger is attached to the running process In the Debug perspective, the Debug view displays the attached debuggers.



Attaching a Debugger to a Running Application

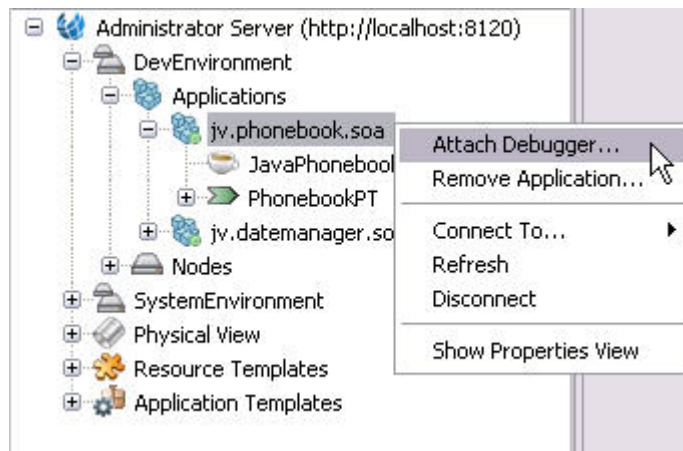
You can attach a debugger to a running application and examine the debug configuration that the debugger creates.

Do not attach a debugger to more than one application running on the same node because they will interfere with each other. To debug multiple applications on the same node, create a remote debug configuration and add multiple applications in the Remote Applications tab.

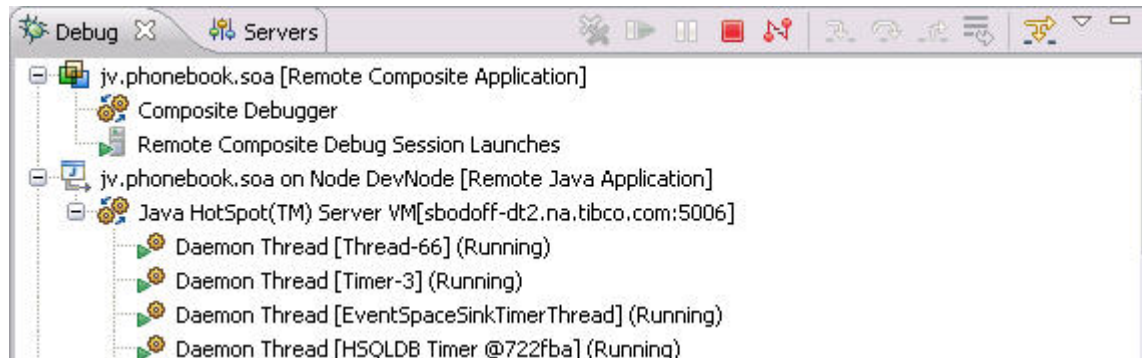
Only one debugger can be attached to an application. The launch configuration tests to see if there is a conflict and if so will display an error. If multiple users try to attach to applications on the same node, the connection will fail.

Procedure

1. In the Administrator Explorer View, expand the nodes of the Administrator server that manages the application.
2. Right-click the application and select **Attach Debugger**



The debugger is attached to the running process and a debug configuration is created if one did not already exist. In the Debug perspective, the Debug view displays the attached debuggers.



TIBCO ActiveMatrix Administrator

TIBCO ActiveMatrix Administrator is the utility used to create, configure, monitor, and manage objects in the TIBCO ActiveMatrix runtime. TIBCO ActiveMatrix Administrator is a web application that provides a browser interface and a command-line scripting interface. The command-line interface is a set of Ant tasks that execute via an automated script.

TIBCO Business Studio provides several features that support interactions with ActiveMatrix Administrator:

- [Deployment Servers view](#) - allows you to create, manage, and connect to Administrator deployment servers.
- [Administrator Explorer View](#) - displays a tree view of the objects managed by a connected Administrator server.
- Administrator CLI scripts generators - allow you to create, deploy, start, stop, undeploy, and delete applications.
 - [Generating an Administrator Command-Line Interface Script](#) - guides you through a step-by-step process to generate an ActiveMatrix Administrator command-line script for a connected or offline ActiveMatrix Administrator server and optionally execute the scripts on a connected server.
 - [Command-Line task](#) - allows you to generate ActiveMatrix Administrator command-line script using a task in a TIBCO Business Studio command-line script.

Deployment Servers

A deployment server defines a connection to a TIBCO ActiveMatrix Administrator server. You manage deployment servers in the Deployment Server view.

Creating a Deployment Server

You can create a Deployment Server in the TIBCO ActiveMatrix Administrator Deployment Server view.

Procedure

1. In the Deployment Server view, right-click **Deployment Servers** and select **New > Server**. The New Server dialog displays.
2. In the Name field, type a name for the server.
3. In the Runtime drop-down list, select **Administrator Server** and click **Next >**. The Runtime Server Parameters dialog displays.
4. Complete the [Runtime Server Parameters](#) dialog.
5. Click **Test Connection**.
6. Click **OK** to dismiss the dialog.
7. Click **Finish**.

Connecting to a Deployment Server

After you have created a Deployment Server, you can connect to it in the Administrator Explorer or Deployment Server view.

Prerequisites

Create a Deployment Server. See [Creating a Deployment Server](#).

Procedure

1. In the Administrator Explorer or Deployment Server view right-click a deployment server, and select **Connect**.
2. If you did not save the password when you created the deployment server, supply the password when prompted.
3. If the server is SSL enabled, accept the certificate when prompted.
4. Supply credentials according to the Administrator configuration:
 - **Password Dialog** - Provide the password and click **OK**.
 - **Certificate Acceptance Dialog** - Click **Yes** to accept the certificate and **Yes** to restart TIBCO Business Studio.

TIBCO Business Studio connects to and retrieves data from the server.

Editing Deployment Server Properties

Procedure

1. In the Deployment Server view, right-click a deployment server and select **Properties**.
2. Edit the properties as described in [Deployment Server Reference](#) dialog.
3. Click **OK**.

Refreshing a Deployment Server

Procedure

- In the Administrator Explorer or Deployment Server view, right-click a deployment server, and select **Refresh**.
The current contents of the deployment server is loaded into the Deployment Server and Administrator Explorer views.

Disconnecting from a Deployment Server

Procedure

- In the Administrator Explorer or Deployment Server view, right-click a deployment server, and select **Disconnect**.

Deployment Server Reference

Field	Description
Server URL	URL of the Administrator server. Default: http://localhost:8120.
Username	Name of the Administrator user who is executing the task. The user must have the permissions required to perform the task.

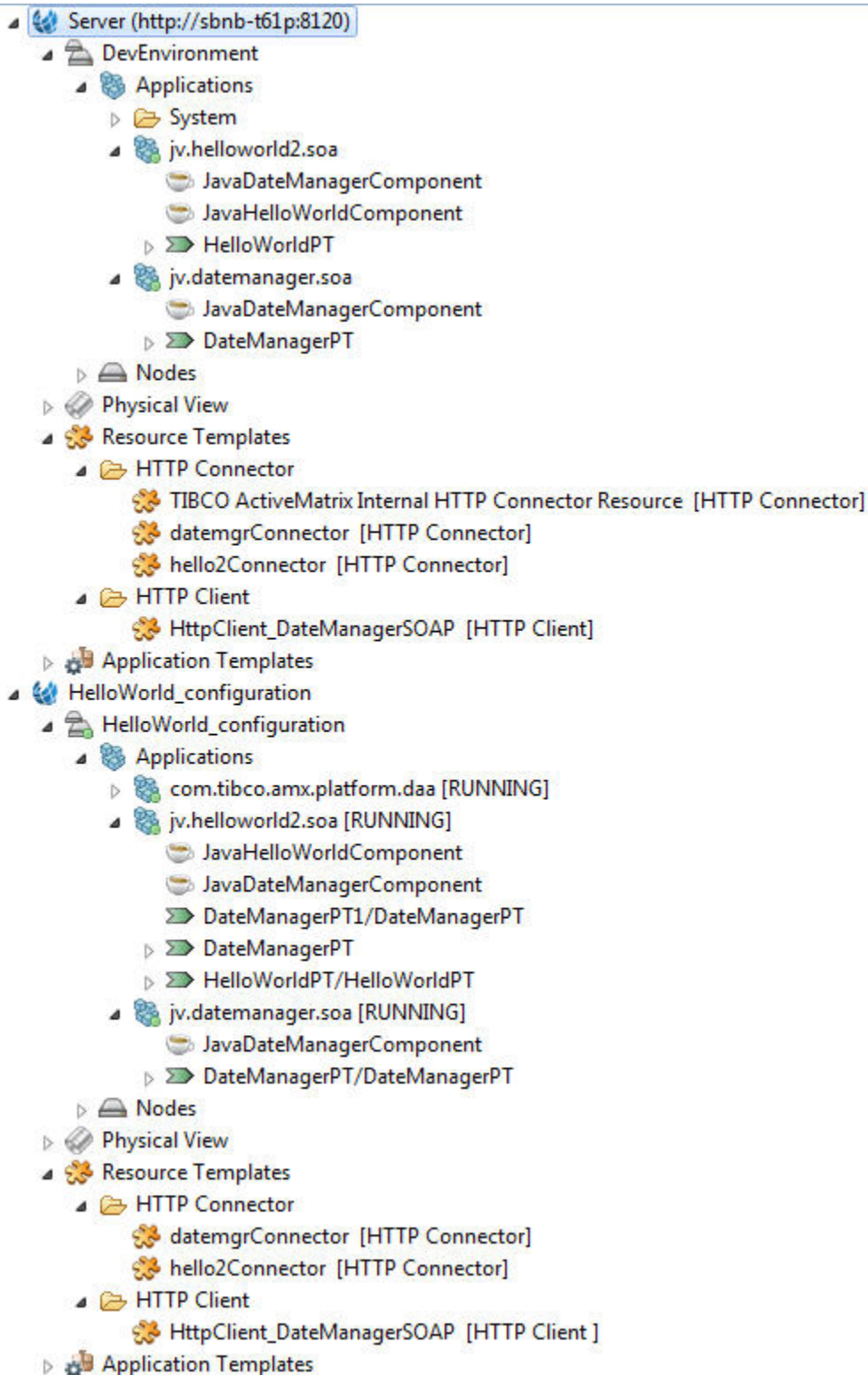
Field	Description
Password	Password of the Administrator user who is executing the task.
Use form-based authentication	After the Administrator server validates the username and password, the server creates a session identified by a unique key that is passed between the client and server on each subsequent HTTP request. Authentication credentials are sent only during the initial handshake and not with every request. This approach improves security.
Hide System Applications	Indicate whether to hide system applications in the Administrator Explorer view.
Default Environment	Default environment to which applications are deployed.
Show Only Default Environment	Indicate whether to only show the default environment in the Administrator Explorer view. After you connect to the deployment server, the Deployment Server view focuses only the selected environment and preselects the default environment in Deploy Module wizard.
Default Application Folder	When you create an application, you can optionally choose a folder to store the application. If you do not choose a folder, the application is stored in the root folder. Permissions assigned to a folder are inherited by all applications contained within the folder.
Default Target Application	The name of the default target application with which applications will be deployed when the distribution policy is Target Application. The field is ignored when the distribution policy is Single Node or Multiple Nodes.
Save Password	Indicate whether to save the password of the deployment server user. The password is obfuscated and stored in a file so that you are not queried for credentials every time you connect to the deployment server.

Administrator Explorer View

The Administrator Explorer view displays a tree view of the remote objects managed by an Administrator server or objects running on a local debug node. The objects include administrative objects such as environments, applications, resource and application templates, and runtime objects such as hosts, nodes, and resource instances.

The format of the objects in the Administrator Explorer view varies depending on whether the objects are running on a remote or local debug node. For example, in [Administrator Explorer View](#), the objects under the `ju.helloworld2.soa` application are slightly different. On the remote node managed by the Administrator server named `Server`, the application shows two components (`JavaDateManagerComponent` and `JavaHelloWorldComponent`) and one service, whereas on the local node managed by the `HelloWorld_configuration` run configuration, the application shows the same two components, the same composite service, and two component services (`DateManagerPT1/DateManager` and `DateManagerPT`).

Administrator Explorer View



You can click objects to display properties of the objects in the Properties view. For example, if you click an application, General and Distribution tabs display in the Properties view. The Distribution tab displays the nodes on which the application is deployed. If you click an application template, General and Instances tabs display in the Properties view.

You can also perform object-specific actions by right-clicking an object. For example, if you right-click a remote application you can attach a debugger to or remove the application. If you right-click a binding you can invoke the Web Services Explorer and [generate a SOAP request](#).



The remote objects in the Administrator Explorer view are imported from the Administrator server when you connect to the server. If you create objects in the Administrator server after you connect, they do not appear in the view. To load the current objects, [refresh](#) the connection.

Displaying the Administrator Explorer View

Prerequisites

Create and connect to a deployment server of type Administrator Server.

Procedure

1. Select **Window > Show View > Other...**
The Show View dialog displays.
2. Select **TIBCO SOA Platform > Administrator Explorer**.

Administrator Command-Line Interface

The Administrator command-line interface (CLI) provides access to most TIBCO ActiveMatrix Administrator functions that change the state of Administrator objects.

You can perform the following actions using the CLI:

- Add, edit, and delete objects
- Install and uninstall
- Start and stop objects
- Set properties and substitution variables
- Distribute application components to nodes

You can use the CLI for repetitive application of standard actions on large numbers of objects.

The CLI is based on the [Ant](#) open source build tool and is implemented in an Ant task named `AMXAdminTask`. You specify the Ant task in a build target within an Ant *build file*. Each instance of `AMXAdminTask` in the build file specifies an action to be performed on one or more objects specified in a *data file*.

The CLI invokes web services exposed by the Administrator server. You specify the Administrator server location and user credentials in a *property file*.

TIBCO Business Studio provides a wizard to generate and execute CLI scripts to perform the following application-related actions in a TIBCO ActiveMatrix Administrator deployment server:

- Upload a DAA
- Add or upgrade feature or node
- Configure and deploy an application
 - Create an application
 - Distribute application fragments to nodes
 - Configure application properties
 - Create resource instances

- Deploy an application
- Start an application
- Undeploy an application
- Delete an application
- Delete old application template version(s)
- Delete old feature version(s)

For information on Ant support in TIBCO Business Studio, see **Help > Help Contents > Workbench User Guide > Concepts > Ant & External tools > Ant support** .

Understanding Build Files

The Ant build file for the command-line interface must contain the `import`, `project`, `target`, and `AMXAdminTask` elements.

import Element

The `import` element identifies the task definition file, which defines the path to the libraries required by `AMXAdminTask`.

Set the `file` attribute to `CONFIG_HOME/admin/amxadmin/samples/admin-scripts-base.xml` . For example:

```
<import file="C:/Documents and Settings/AMX-User/ApplicationData/amx-3/data/admin/
amxadmin/samples/admin-scripts-base.xml"/>
```

project Element

The `project` element declares the default build target for the `build.xml` file. `taskdef` and `target` are subelements of the `project`. The optional `default` attribute allows you to specify a default target. You can choose any target from the build file to be the default target.

```
<project default="target">
  <taskdef ... />
  <target name="target" ... />
</project>
```

target Element

The `target` element specifies the actions performed for an execution of the command line interface via the `AMXAdminTask` subelement. In a target you can provide a `depends` attribute containing a list of targets. Each target will be run in order until one fails or the list completes.

```
<target name="target">
  <AMXAdminTask ... />
</target>
```


Example Build File

The following build file defines targets to upload a distributed application archive, create an application, map an application to a node, create a resource template, create a resource instance and install it in a node, and deploy an application.

```
<project default="all">

  <dirname property="admin.samples.directory" file="CONFIG_HOME/
admin/enterpriseName/samples"/>

  <!-- This import defines the custom AMXAdminTask. -->
  <import file="${admin.samples.directory}/admin-scripts-
base.xml"/>

  <!-- Predefine ${dataFile} to apply the targets in this script
with different parameters. -->
  <property name="dataFile" value="userProvided dataFile"/>

  <!-- Predefine ${instanceProperties} to control a different
Administrator server with this script. -->
  <property name="remote-properties.file" value="$
{admin.samples.directory}/remote_props.properties"/>

  <!-- Default task for this build file -->
  <target name="all"
    depends="upload.daa, create.app, edit.properties,
wire.application, distribute.app,
    deploy.app, start.app"
    description="Default target group, execute following
targets: upload.daa, create.app,
    edit.properties, wire.application, distribute.app, deploy.app,
start.app"/>

  <!-- Upload DAA specified in the data file -->
  <target name="upload.daa" description="Uploading Application">
    <AMXAdminTask
      action="add"
      objectSelector="DAA"
      remote="true"
      propsFile="${remote-properties.file}"
      dataFile="${dataFile}"
      overwrite="false" merge="true" createIfNotExists="true"
      force="false" failOnError="false" />
  </target>

  <!-- create the application -->
  <target name="create.app" description="Creating Application">
    <AMXAdminTask remote="true" propsFile="${remote-
properties.file}"
      action="add" dataFile="${basedir}/
jv.phonebook.soa.deployment-config.xml"
      objectSelector="Environment//Application"
      overwrite="false" merge="true"
      createIfNotExists="true" force="false"
failOnError="true" />
  </target>

  <!-- configure properties of the application, and create
resource instances if needed -->
  <target name="edit.properties" description="Editing Properties">

    <!-- create resource template -->
    <AMXAdminTask remote="true" propsFile="${remote-
properties.file}"
      action="add" dataFile="${dataFile}"
      objectSelector="ResourceTemplate" overwrite="false"
merge="true"
```

```

        createIfNotExists="true" force="false"
failOnError="true" />

        <!-- add all require resource instances -->
        <AMXAdminTask remote="true" propsFile="${remote-
properties.file}"
            action="add" dataFile="${dataFile}"
            objectSelector="Environment/Node/ResourceInstance"
            overwrite="false" merge="true"
            createIfNotExists="true" force="false"
failOnError="true" />

        <!-- install instances added above -->
        <AMXAdminTask remote="true" propsFile="${remote-
properties.file}"
            action="install" dataFile="${dataFile}"
            objectSelector="Environment/Node/ResourceInstance"
            overwrite="false" merge="true"
            createIfNotExists="true" force="false"
failOnError="true" />

        <!-- override values for properties -->
        <AMXAdminTask remote="true" propsFile="${remote-
properties.file}"
            action="edit" dataFile="${dataFile}"
            objectSelector="Environment//Application/Property |
                Environment//Application//PromotedService//Binding/
Property |
                Environment//Application//PromotedReference//Binding/
Property"
            overwrite="false" merge="true"
            createIfNotExists="true" force="false"
failOnError="true" />

    </target>

    <!-- create wires to other applications -->
    <target name="wire.application" description="Wiring Application">
        <AMXAdminTask remote="true" propsFile="${remote-
properties.file}"
            action="set" dataFile="${dataFile}"
            objectSelector="//PromotedReference/Wire"
            overwrite="false" merge="true"
            createIfNotExists="true" force="false"
failOnError="true" />
    </target>

    <target name="distribute.app" description="Distributing
Application">
        <AMXAdminTask
            action="set"
            objectSelector="Environment//Application//Component/Node
|
                Environment//Application//PromotedService//Binding/
Node |
                Environment//Application//PromotedReference//Binding/
Node"
            remote="true"
            propsFile="${remote-properties.file}"
            dataFile="${dataFile}"
            overwrite="false"
            merge="true"
            createIfNotExists="true"
            force="false"
            failOnError="false"/>

    </target>

    <!-- deploy the application -->

```

```

    <target name="deploy.app" description="Deploying Application">
      <AMXAdminTask remote="true" propsFile="${remote-
properties.file}"
        action="deploy" dataFile="${dataFile}"
        objectSelector="Environment//Application"
        overwrite="false" merge="true"
        createIfNotExists="true" force="false"
failOnError="true"
      />
    </target>

    <target name="start.app" description="Starting Application">
      <AMXAdminTask remote="true" propsFile="${remote-
properties.file}"
        action="start" dataFile="${dataFile}"
        objectSelector="Environment//Application"
        overwrite="false" merge="true"
        createIfNotExists="true" force="false"
failOnError="true" />
    </target>
  </project>

<project default="all">
  <dirname property="admin.samples.directory" file="CONFIG_HOME/
admin/enterpriseName/samples"/>

  <!-- This import defines the custom AMXAdminTask. -->
  <import file="${admin.samples.directory}/admin-scripts-
base.xml"/>

  <!-- Predefine ${dataFile} to apply the targets in this script
with different parameters. -->
  <property name="dataFile" value="userProvided dataFile"/>

  <!-- Predefine ${instanceProperties} to control a different
Administrator server with this script. -->
  <property name="remote-properties.file" value="$
{admin.samples.directory}/remote_props.properties"/>

  <target name="all" depends="upload.daa, create.app,
map.app.to.node, create.rt,
create.ri, install.ri, deploy.app"/>

  <target name="upload.daa">
    <AMXAdminTask
      propsFile="${remote-properties.file}"
      action="add"
      dataFile="dateMgr_data.xml"
      objectSelector="DAA"
      failOnError="true"/>
  </target>

  <target name="create.app">
    <AMXAdminTask
      remote="true"
      propsFile="${remote-properties.file}"
      action="add"
      dataFile="dateMgr_data.xml"
      objectSelector="Environment//Application"
      failOnError="true"/>
  </target>

  <target name="map.app.to.node">
    <AMXAdminTask
      remote="true"
      propsFile="${remote-properties.file}"
      action="set"
      dataFile="dateMgr_data.xml"
      objectSelector="Environment//Application/Node"
      failOnError="true"/>
  </target>

```

```

</target>

<target name="create.rt">
  <AMXAdminTask
    remote="true"
    propsFile="${remote-properties.file}"
    action="add"
    dataFile="dateMgr_data.xml"
    objectSelector="ResourceTemplate"
    failOnError="true"/>
</target>

<target name="create.ri">
  <AMXAdminTask
    remote="true"
    propsFile="${remote-properties.file}"
    action="add"
    dataFile="dateMgr_data.xml"
    objectSelector="Environment/Node/ResourceInstance"
    failOnError="true"/>
</target>

<target name="install.ri">
  <AMXAdminTask
    remote="true"
    propsFile="${remote-properties.file}"
    action="install"
    dataFile="dateMgr_data.xml"
    objectSelector="Environment/Node/ResourceInstance"
    failOnError="true"/>
</target>

<target name="deploy.app">
  <AMXAdminTask
    remote="true"
    propsFile="${remote-properties.file}"
    action="deploy"
    dataFile="dateMgr_data.xml"
    objectSelector="Environment//Application"
    failOnError="true"/>
</target>

</project>

```

Understanding AMXAdminTask

AMXAdminTask specifies an action, data and property files, the objects on which the action is performed, and various behavioral attributes.


```

<AMXAdminTask
  action="action"
  dataFile="path to data file"
  propsFile="path to properties file"
  [createIfNotExists = "{true|false}"]
  [failOnError="{true|false}"]
  [force="{true|false}"]
  [merge="{true|false}"]
  [objectSelector="XPath expression"]
  [options="nostart|immediate|terminate|resolve|auto-resolve|stable|
handleDependencies"]
  [overwrite="{true|false}"]
  [timeout="timeout value"/>

```

Parameters

Attribute	Type	Req?	Description
action	String	Yes	<p>The action to be performed on the objects in the data file. The valid actions are:</p> <ul style="list-style-type: none"> • add • edit • install • uninstall • start • stop • deploy • undeploy • delete • remove <p>The action is case insensitive.</p> <p>Unless <code>objectSelector</code> is specified, the action is applied to every object in the data file.</p> <p>The order in which the action is applied to the objects is either breadth first or depth first. The method used is determined by the action.</p> <ul style="list-style-type: none"> • Breadth first - add, edit, install, start, stop • Depth first - delete, uninstall <p>Some actions are not performed against certain object formats.</p> <ul style="list-style-type: none"> • For the most part, add and edit are applied only to objects specified in full format. Objects not in this format are skipped.
createIfNotExists	Boolean	No	<p>Applicable to the edit action.</p> <p>If an object is to be edited but doesn't yet exist and this flag is true, then the object is added.</p> <p>If this flag is false and the object to be edited doesn't exist, an error is reported.</p> <p>Default: true.</p>
dataFile	String	Yes	<p>The path to the XML file containing the object data.</p>

Attribute	Type	Req?	Description
failOnError	Boolean	No	<p>Causes the Ant task to fail when an unrecoverable error is reported. The option stops processing of targets in the depends list or specified on the command line.</p> <p>Default: true</p>
force	Boolean	No	<p>Forces an action even if the object has dependent objects or is not in the appropriate state. Applies to the following actions and objects:</p> <ul style="list-style-type: none"> • delete - Node, Application, Environment, ResourceTemplate, ResourceInstance • undeploy - Application • stop - Application, Component, Binding • uninstall - Node, ResourceInstance <p>For example:</p> <ul style="list-style-type: none"> • A node must be in the uninstalled state before it can be deleted and it must be stopped before it can be uninstalled. If any problems occur moving the node to one of these states, and force is true, the node is deleted even if it is not in the uninstalled state or uninstalled even if it is not stopped. • An application must be in the undeployed state before it can be deleted and it must be stopped before it can be undeployed. If any problems occur moving the application to one of these states, and force is true, the application is deleted even if it is not in the undeployed state. <div>  <p>You should exercise extreme caution when using this option as it may leave your system in a non-working state.</p> </div> <p>Default: false.</p>

Attribute	Type	Req?	Description
merge	Boolean	No	<p>Applicable to the add action, and only if the overwrite flag was used and is true.</p> <p>If an object to be added already exists and</p> <ul style="list-style-type: none"> • If merge is true and overwrite is true, then the existing object is overwritten by merging with the new object. That is, the old object's data is updated with the new object's data. • If merge is false but overwrite is true, then the existing object is deleted and replaced by the new object. The old object's children and access control lists, if any, are lost in the process. <p>Default: true.</p>
objectSelector	String	No	<p>Specifies the set of objects to be processed by an XPath expression. For information on the XPath language, see http://www.w3.org/TR/xpath. If this attribute is not specified:</p> <ul style="list-style-type: none"> • All of the objects in the data file are processed. • The heuristic used to determine the order in which the objects are processed depends on the action option.

Attribute	Type	Req?	Description
options	String	No	<p>The valid options are:</p> <ul style="list-style-type: none"> auto-resolve Applies to: install action when installing resource instances Causes the node to be re-started if needed. handleDependencies Applies to: deploy, undeploy, start, and stop actions <ul style="list-style-type: none"> For the deploy action, this causes optPreFlight to be performed before the deployment. For information, see "Deployment Preflight Check" in <i>TIBCO ActiveMatrix BPM SOA Administration</i>. For the undeploy action, this causes optPreFlight to be performed before the undeployment. For information, see "Undeployment Preflight Check" in <i>TIBCO ActiveMatrix BPM SOA Administration</i>. For the start action, this causes startPreFlight to be performed before starting the application. For information, see "Starting Applications" in <i>TIBCO ActiveMatrix BPM SOA Administration</i>. For the stop action, this causes stopPreFlight to be performed before starting the application. For information, see "Stopping Applications" in <i>TIBCO ActiveMatrix BPM SOA Administration</i>. immediate Applies to: stop action Causes applications, components, bindings, and nodes to perform a quick cleanup and then stop. nostart Applies to: deploy action Prevents applications from being started after deployment. resolve Applies to: deploy, undeploy, install, add, remove actions

Attribute	Type	Req?	Description
			<p>Causes nodes to be restarted when a node is installed, a feature is added or removed from a node, or an application is deployed or undeployed from a node.</p> <ul style="list-style-type: none"> • stable Applies to: <code>install</code> action This is available when installing resource instances. It prevents the nodes from restarting. In this mode, what you deploy should not affect any other running code in the runtime. • terminate Applies to: <code>stop</code> action This applies only to nodes and causes the node process to be killed without any cleanup.
<code>overwrite</code>	Boolean	No.	<p>Applicable to the <code>add</code> action. If an object to be added already exists and the <code>overwrite</code> is true, then the existing object is overwritten.</p> <p>There are two ways in which an object can be overwritten: it can be merged, or created from scratch. The strategy used is determined by the <code>merge</code> option.</p> <p>Default: true.</p>
<code>propsFile</code>	String	Yes	<p>The path to the properties file containing the Administrator server location and user-specific information data.</p>
<code>skipIfNotExists</code>	Boolean	No	<p>Used when deleting an object.</p> <p>When set to true, no attempt is made to delete the object if it does not exist.</p> <p>When set to false, an error is reported if the object to be deleted does not exist.</p> <p>Default: false.</p>

Attribute	Type	Req?	Description
timeout	Integer	No	<p>Length of time in seconds that a target will wait for an action to complete before reporting an error. If a timeout occurs and <code>failOnError</code> is true, the Ant task will fail. If a timeout occurs and <code>failOnError</code> is false, the script will report an error but the script will continue to process targets.</p> <p>This option applies only to the following asynchronous actions and objects:</p> <ul style="list-style-type: none"> • <code>deploy</code>, <code>undeploy</code> - Application, Plug-in • <code>install</code> and <code>uninstall</code> - Node, ResourceInstance • <code>start</code> - Node <p>Default: 0, which means the task will never time out. You should not change the default unless you are creating large amounts of data and leaving the script run unattended or have a requirement that node startup satisfies a timing constraint.</p>

create

Assume you have an environment `env1` in the database. Your data file has environment `env1` and a node `node1`. If you specify the edit action and

- `createIfNotExists` = false. `env1` already exists, so its data is edited to match `env1` in the data file. `node1` doesn't exist, so is not updated.
- `createIfNotExists` = true. `env1` already exists, so its data is edited to match `env1` in the data file. `node1` doesn't exist, so it is added to `env1`.

force

Assume you have an environment `env1` and node `node1` in both the database and the data file. `node1` is in the Started state. If you do a delete and

- `force` = false. `node1` is in the Started state. There are two possible outcomes:
 - The stop and uninstall are successful. `node1` and `env1` are deleted.
 - The stop or uninstall fails. `node1` is not in the uninstalled state so it cannot be deleted. The delete does not complete.
- `force` = true. `node1` is in the Started state. There are two possible outcomes:
 - The stop and uninstall are successful. `node1` is deleted. `env1` is deleted.
 - The stop or uninstall fails. `node1` is not in the uninstalled state but is forcefully deleted. `env1` is deleted.

objectSelector

- objectSelector="//*"

Process all objects.
- objectSelector="//Node"

Process all nodes.
- objectSelector="/Environment[@name='env1']/Node[@name='node1']"

Process node1 in environment env1.

overwrite and merge

Assume you have environment env1 and node2 in the database. If you specify the add action with a data file that contains env1 and node1:

- overwrite = false (merge is then ignored). Nothing happens to env1. node1 is added.
- overwrite = true and merge = false. env1 is deleted and replaced with the env1 in the data file and node2 is deleted. node1 doesn't exist yet and is added.
- overwrite = true and merge = true. The existing env1 is updated with data from the env1 in the data file. Nothing happens to node2 and node1 is added.

Actions Performed Using CLI

The actions that can be performed with the command-line interface affect either the objects contained in the database or the objects executing in the TIBCO ActiveMatrix runtime.

Database Actions

Database actions modify the objects contained in the Administrator database:

- add - Add an object or an association between objects, such as between an application and a node.
- addOrUpgrade -
 - Adds the features from your data file to the nodes. For example, Feature `com.acme.myapp.feature 1.5.0` gets added to DevNode with status `Marked for Install`.
 - Removes any other versions of the same feature, if they were present on the nodes. Features with a different major version are not affected. For example, Feature `com.acme.myapp.feature 1.0.0` and `1.6.0` are changed to status `Marked for Uninstall`, since any version not matching "1.5.0" will be marked for removal. However, feature `com.acme.myapp.feature 2.0.0` is left as is, because it has a different major version.
- edit - Edit an object.
- delete - Delete an object or an association between objects. When you delete an object, the entire tree rooted at the object is deleted starting at the leaves.
- set - Set the value of a substitution variable, map an application, component or binding to a node, set a property of a binding. This action deletes any existing entries that aren't present in the new set and adds any entries in the new set that weren't in the database.
- upgrade - Upgrade an existing application.
- promote - Make a service or reference available at the environment level for cross-environmental wiring.

- demote - Make a service or reference unavailable at the environment.
- resetPassword - Reset a user password.

Runtime Actions

Runtime actions modify the state of the objects contained in the TIBCO ActiveMatrix runtime:

- install - Install node on a host or a resource instance on a node.
- uninstall - Uninstall a node from a host or a resource instance from a node.
- deploy - Deploy a component or binding to a node, a logging configuration for a host, node, application, or component, a plug-in to the Administrator server. Also undeploys components and bindings from nodes they are no longer mapped to.
- undeploy - Undeploy an application or plug-in.
- start - Start a node, application, component, or binding.
- stop - Stop a node, application, component, or binding.

Understanding Data Files

The data file is an XML file that specifies attributes of the objects that are operated on by AMXAdminTask.

A data file has the following structure.

```
<amxdata_base:Enterprise
xmlns:amxdata="http://tibco.com/amxadministrator/command/line/types"
xmlns:amxdata_base="http://tibco.com/amxadministrator/command/line/types_base"
xmlns:amxdata_reference="http://tibco.com/amxadministrator/command/line/
types_reference"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://tibco.com/amxadministrator/command/line/types ../schemas/
amxdata.xsd
http://tibco.com/amxadministrator/command/line/types_base ../schemas/
amxdata_base.xsd
http://tibco.com/amxadministrator/command/line/types_reference ../schemas/
amxdata_reference.xsd">

    Objects

</amxdata_base:Enterprise>
```

Example Data File

The following is a sample of the data file:

```
<amxdata_base:Enterprise xmlns:amxdata="http://tibco.com/
amxadministrator/command/line/types" xmlns:amxdata_base="http://
tibco.com/amxadministrator/command/line/types_base"
xmlns:amxdata_reference="http://tibco.com/amxadministrator/command/
line/types_reference" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://tibco.com/amxadministrator/
command/line/types platform:/plugin/com.tibco.amf.tools.admincligen/
model/cli_data.ecore#//types http://tibco.com/amxadministrator/
command/line/types_base platform:/plugin/
com.tibco.amf.tools.admincligen/model/cli_data.ecore http://
tibco.com/amxadministrator/command/line/types_reference platform:/
plugin/com.tibco.amf.tools.admincligen/model/cli_data.ecore#//types/
types_reference">
  <Environment name="DevEnvironment" xsi:type="amxdata:Environment">
    <Node name="DevNode" xsi:type="amxdata:Node">
      <ResourceInstance applicationName="TestCLI"
name="NewJDBCResource" resourceTemplateName="NewJDBCResource"
scopeType="Application" xsi:type="amxdata:ResourceInstance"/>
      <ResourceInstance applicationName="TestCLI"
name="NewJmsConnFactoryResource"
resourceTemplateName="NewJmsConnFactoryResource"
scopeType="Application" xsi:type="amxdata:ResourceInstance"/>
      <ResourceInstance applicationName="TestCLI"
name="NewSMTPResource" resourceTemplateName="NewSMTPResource"
scopeType="Application" xsi:type="amxdata:ResourceInstance"/>
      <ResourceInstance applicationName="TestCLI"
name="httpConnector" resourceTemplateName="httpConnector"
scopeType="Application" xsi:type="amxdata:ResourceInstance"/>
    </Node>
    <Node name="DevNode" xsi:type="amxdata:Node">
      <Feature componentID="TestCLI.customfeature.id"
version="1.0.0" xsi:type="amxdata_base:FeatureID"/>
      <Feature componentID="TestCLI.customfeature.id"
version="1.0.0.v2017-03-15-1808" xsi:type="amxdata_base:FeatureID"/>
    </Node>
    <Application folderPath="/" importResourceTemplates="true"
name="TestCLI" resourceTemplatesScope="application"
xsi:type="amxdata:Application">
      <Component name="Component1"
xsi:type="amxdata_base:Component_base">
        <Node environmentName="DevEnvironment" name="DevNode"
xsi:type="amxdata_reference:Node_reference"/>
      </Component>
      <Component name="Mediation1"
xsi:type="amxdata_base:Component_base">
        <Node environmentName="DevEnvironment" name="DevNode"
xsi:type="amxdata_reference:Node_reference"/>
      </Component>
      <Property name="Component1_Property1"
propertyType="JdbcDataSource" value="NewJDBCResource"
xsi:type="amxdata:Property"/>
      <Property name="Component1_Property2"
propertyType="SmtpConfiguration" value="NewSMTPResource"
xsi:type="amxdata:Property"/>
      <Property name="Component1_Property3"
propertyType="JMSConnectionFactory"
value="NewJmsConnFactoryResource" xsi:type="amxdata:Property"/>
      <Property name="MEDIATION_VALIDATE_MESSAGE_DATA"
propertyType="boolean" value="false" xsi:type="amxdata:Property"/>
      <PromotedService name="HelloWorldPT"
xsi:type="amxdata_base:Service_base">
        <Binding name="Binding00"
xsi:type="amxdata_base:Binding_base">
          <Node environmentName="DevEnvironment" name="DevNode"
xsi:type="amxdata_reference:Node_reference"/>
          <Property name="HttpInboundConnectionConfig"
propertyType="HttpConnector" value="httpConnector"
```

```

xsi:type="amxdata:Property"/>
    </Binding>
    </PromotedService>
    <ResourceTemplate headerBufferSize="4096" host="0.0.0.0"
name="httpConnector" port="9090" requestBufferSize="8192"
xsi:type="amxdata:HttpConnectorResourceTemplate"/>
    <ApplicationTemplate name="TestCLI"
version="1.0.0.v2017-03-15-1808"
xsi:type="amxdata_reference:ApplicationTemplate_reference"/>
    <ImportResourceTemplateName>NewJDBCResource</
ImportResourceTemplateName>
    <ImportResourceTemplateName>NewSMTPResource</
ImportResourceTemplateName>
    <ImportResourceTemplateName>NewIndiConnResource</
ImportResourceTemplateName>
    <ImportResourceTemplateName>NewJmsConnFactoryResource</
ImportResourceTemplateName>
    </Application>
    </Environment>
    <DAA location="/Users/macbookkwan/workspace_testintaller_delta_2/
TestCLI/Deployment Artifacts/TestCLI.daa" xsi:type="amxdata:DAA">
    <importFeatureIdentifier>TestCLI.customfeature.id:
1.0.0.v2017-03-15-1808</importFeatureIdentifier>
    </DAA>
    <Feature componentID="TestCLI.customfeature.id"
version="1.0.0.v2017-03-15-1808" xsi:type="amxdata_base:FeatureID"/>
    <AppTemplate name="TestCLI" version="1.0.0"
xsi:type="amxdata_base:AppTemplateID"/>
</amxdata_base:Enterprise>

```

Understanding Objects

You specify the objects on which the command-line interface operates in an XML data file. TIBCO ActiveMatrix Administrator provides XSD schemas for the data files that capture all of the ID attributes, description attributes, parent-child relationships, and associative relationships of objects.

Every object is described in an XML element. The attributes of that object (both ID and descriptive) are attributes of the XML element, and the relationships this object has with other objects are subelements of the XML element. In these schemas, every Administrator object can be specified in three types of formats: base, full, and reference.

Object Schemas

The object schemas are located in *TIBCO_HOME/administrator/version/schemas* and are named:

- *amxdata.xsd* Full format definitions.
- *amxdata_base.xsd* Base format definitions.
- *amxdata_reference.xsd* Reference format definitions.
- *amxdata_detailed.xsd* Currently this schema is empty. It is reserved for use in the future.

Object Formats

Objects in the data XML file of a CLI script can be specified in three formats: base, full, and reference.

Base Format

Base format uniquely identifies the object. Base format is defined in the schema *amxdata_base.xsd*. The base format is a convenience so that you do not have to give all the descriptive attributes of an object to work with it. Base format:

- Captures the ID attributes of an object as XML attributes
- Captures the parent-child relationships of an object as XML elements

- Doesn't capture any parent information about the object as that information is derived from the XML structure

You use the base format to:

- Delete an object
- Perform a runtime action on an object
- Add a child to an object
- Perform an action on a child of an object

Full Format

Full format is derived from the base format and includes all the base format information plus additional attributes that describe the objects. Full format is defined in the schema `amxdata.xsd`. Full format:

- Is derived from base format
- Captures the ID and description attributes of an object as XML attributes
- Captures the parent-child and associative relationships of an object as XML elements
- Doesn't capture any parent information about the object as that information is derived from the XML structure

You use full format:

- Whenever the base format can be used
- To add or edit an object

Reference Format

Reference format is used for making associations between two objects. Reference format is defined in the schema `amxdata_reference.xsd`. Reference format:

- Captures the ID attributes of an object as XML attributes
- Objects not residing directly under the Enterprise object have parent information because it cannot be derived from the XML structure

You use reference format:

- When associating that object to another object

Property File Reference

Property files contain Administrator server location and user-specific information used when running the command-line interface.

Property	Type	Description
adminURL	URL	URL of the Administrator server.
username	String	Name of the Administrator user executing the task. The user must have the permissions required to execute the actions in the script.

Property	Type	Description
password	String	Password of the Administrator user executing the task.
httpConnectionTimeout (s)	Integer	Length of time to wait for the connection to the Administrator server to establish.
httpAuthType	String	<p>Type of authentication to use to secure communication between the Administrator CLI and the remote Administrator server:</p> <ul style="list-style-type: none"> • basic - Use basic authentication. The username and password credentials are sent in each request. Before transmission, the user name is appended with a colon and concatenated with the password. The resulting string is encoded with the Base64 algorithm. • form - Use form-based authentication. After the username and password credentials are validated by the Administrator server, the server creates a session identified by a unique key that is passed between the client and server on each subsequent HTTP request. This approach is more secure because authentication credentials are only sent during the initial handshake and not with every request. <p>Default: basic.</p>
javax.net.ssl.trustStore	String	Trust store properties used by the Administrator CLI to connect to the Administrator server when the external HTTP connector is enabled with SSL. The property values are used to create the trust store file in the location specified by the <code>javax.net.ssl.trustStore</code> property.
javax.net.ssl.trustStoreType	String	
javax.net.ssl.trustStorePassword	Obfuscated password	
admin.cli.ssl.keystorelocation	String	Keystore properties are used by the Administrator CLI to connect to the TIBCO Host instance when it is enabled with JMX over SSL. The property values are used to create the keystore file in the location specified by the
admin.cli.ssl.keystorepassword	Obfuscated password	
admin.cli.ssl.keystoretype	String	

Property	Type	Description
admin.cli.ssl.keyalias	String	admin.cli.ssl.keystorelocation property.
admin.cli.ssl.keypassword	Obfuscated password	

Generating an Administrator Command-Line Interface Script

You generate an Administrator command-line interface script using the Generate Administrator CLI Script... wizard or using the `sds.generateDeploymentScript` task in a TIBCO Business Studio Command-Line Interface script.

Procedure

1. Right-click a DAA and select **Generate Administrator CLI Script**.
2. Complete the wizard screens and click **Next** to advance to each screen.
 - a) [Script Configuration](#)
 - b) [Administrator Connection Configuration](#)
 - c) [Application Configuration](#)
 - d) [Distribution](#)
 - e) [Property Configuration](#)
 - f) [Wiring Configuration](#)
3. Click **Generate**.
The following files are generated in the destination folder:
 - An Ant build file named `DAAName.deployment-build.xml`, which contains targets that invokes actions defined by the TIBCO ActiveMatrix Administrator CLI. The default target of the script is `all`.
 - A data file named `DAAName.deployment-config.xml`, which contains the object definitions on which the build script invokes actions.
 - If you choose to tokenize the script, a properties file named `DAAName.deployment-config.properties` containing object attributes that can be updated each time you run the build file.
 - A [property](#) file named `ServerName.properties` containing the Administrator server properties required to execute the Administrator CLI script.

If you chose to execute the script, a screen displays with the results. If you selected **Offline Server** in the Administrator Connection Configuration screen, execution will fail. If you leave the **Open Administrator Explorer View** checkbox checked, the Administrator Explorer View displays after execution completes. If you leave the **Close Dialog** checkbox checked, the summary dialog closes when execution completes.
4. Optionally, click **Cancel Execution** to stop execution of the script.

Script Configuration

You configure the default action, actions in build file, destination folder, execute generated script, and tokenize script properties for your script.

Modify the following fields to change the script.

Field	Description
Default Action	<p>The default action to perform in the script:</p> <ul style="list-style-type: none"> • Start Application • Upload DAA - If the DAA does not contain an application template, only the Upload DAA action is available. • Configure and Deploy Application • Undeploy Application • Delete Application <p>Default: Start Application.</p>
Actions in build file	<p>The Administrator CLI actions in the build file. Optionally uncheck tasks to exclude them from the script.</p> <p>Default: Varies depending on the choice in Default Action.</p>
Destination Folder	<p>The folder to contain the generated script files.</p> <p>Default: The folder containing the DAA.</p>
Execute Generated Script	<p>Indicate whether to execute the script on completing the wizard.</p> <p>Default: Unchecked.</p>
Tokenize Script	<p>Indicate whether some object attributes, such as the deployment environment and node names, application name and version, and property values, should be externalized into a properties file so that you can optionally update them each time you run the build file.</p> <p>Default: Unchecked.</p>

Administrator Connection Configuration

The Administrator connection configuration specifies the connection to the Administrator deployment server. You can modify the configuration by changing fields that describe the server and its attributes. You can add a new server as part of configuration.

Configure the connection to the Administrator deployment server. The connection configuration is stored in a [property](#) file named *ServerName.properties*.

Field	Description
Online Server	<p>Indicate that an online server should be used in the CLI script. When selected, the Server Name and Add a new server... fields are enabled. The Server Name drop-down list contains the available servers. If there is no servers in the list, click Add a new server...</p> <p>Default: Online Server</p>
Server Name	The name of an Administrator deployment server.
Last Updated	The length of time since the server connection was updated. Click Refresh to update the connection.

Field	Description
Add a new server	Invokes the Creating a Deployment Server wizard.
Offline Server	Indicate that an offline server should be used in the CLI script. When selected the Administrator URI, Username, Password, Use Form-Based Authentication, and Server Version fields are enabled.
URL	The URL of the offline server. Default: http://localhost:8120.
Username	Name of the Administrator user who is executing the task. The user must have the permissions required to execute the actions in the script. Default: root.
Password	Password of the Administrator user executing the task. Default: t.
Use Form-Based Authentication	Use form-based authentication. After the username and password credentials are validated by the Administrator server, the server creates a session identified by a unique key that is passed between the client and server on each subsequent HTTP request. This approach is more secure because authentication credentials are only sent during the initial handshake and not with every request.
Server Version	Version of the Administrator deployment server for which the script is being generated: <ul style="list-style-type: none"> Unknown Version 3.1.x 3.2.0 3.3.0 Default: 3.3.0

Application Configuration

Application configuration allows you to configure application properties. The fields display if the DAA contains an application template.

Field	Description
Environment Name	The name of the environment in which to create the application. If connected to an online server, select an environment. If using an offline server, type an environment name. Default: None.
Application Name	The name of the application. Application names cannot contain the characters \, /, :, *, ?, ", <, >, , whitespace, %, #, &, (,), or comma. Default: The name of the DAA.

Field	Description
Upgrade Existing Application	Indicate whether to upgrade the application if it already exists. For an online server, the checkbox is enabled and checked when the application already exists and disabled if the application does not exist. For an offline server it is always enabled. When the checkbox is checked, the wizard generates an upgrade.app target that invokes the Administrator command-line interface upgrade action. When the checkbox is unchecked, the wizard generates a create.app target that invokes the Administrator command-line interface add action.
Re-provision Custom Feature(s)	Applies any feature changes (additions or removals) to the runtime node. It is the equivalent of clicking Apply on the Node > Configuration > Features screen. If the features are already in-sync, this action does nothing. Additionally, specifying the attribute <code>options="resolve"</code> , rewires the existing features along with the changed features and then restarts the node. The resolve mode is sometimes necessary for feature upgrades or downgrades.
Contact	Optional contact information. Default: None.
Description	Optional description information. Default: None.
Timeout	The length of time in seconds that the script will wait for an action to complete before reporting an error. Default: 0.
Resolve Mode	Indicate whether the latest version of the application template should be used when the application is deployed.

Distribution

You can configure whether to distribute the application fragments contained in the application template on a single node or on multiple nodes. You can also specify the name of the application.

Field	Description
Single Node	The node on which to deploy all application fragments.
Multiple Nodes	Indicate that you want to specify the distribution of each application fragment. If you specify a node in a parent object, it is propagated to all the child objects. <ol style="list-style-type: none"> When you click Next a list of artifact to node mappings displays. For each artifact, click the Nodes column. Specify a node: <ul style="list-style-type: none"> Click Ctrl+Space to display a list of available nodes and double-click a node. Type a comma-separated list of one or more nodes. Click outside a cell.

Property Configuration

You can configure the Substitution Variable and Resource Instance properties of the objects contained in the application template.

The properties display in a table view.

Name	Type	Value
[-] Application Properties		
[-] database1	JdbcDataSource	hsqldb (New Resource from Template hsqldb in DAA)
[-] Binding Properties		
[-] PhonebookPT		
[-] HttpInboundConnectionConfig	string	phonebookConnector

Click icons below the table to toggle between a tree  and table  view of the properties. Accept the configuration or click a row and click **Override** to change the configuration.

Property Type	Procedure
Substitution Variable	Check the Map through Substitution Variable checkbox and type the name of a substitution variable in the Substitution Variable field.

Property Type	Procedure
Resource Instance	<div data-bbox="486 306 528 348"></div> <p data-bbox="598 268 1460 394">If you choose to use an offline server, the Existing Resource Instance and New Resource Instance from Existing Template drop-down lists will be empty. You can type the names of the required resource templates and instances.</p> <ul style="list-style-type: none"> <li data-bbox="486 426 973 457">• (Composite Property Only) No Value <ol style="list-style-type: none"> <li data-bbox="526 489 1436 548">1. Set empty value for a composite property if a resource instance does not exist. <li data-bbox="486 567 853 598">• Existing Resource Instance <ol style="list-style-type: none"> <li data-bbox="526 627 1436 659">1. In the Name drop-down list, select the resource instance or type a name. <li data-bbox="486 674 1465 732">• (Not available for HTTP Connector) New Resource Instance from Template in DAA <ol style="list-style-type: none"> <li data-bbox="526 764 1388 823">1. In the Template Name drop-down list, choose the template or type a template name. <li data-bbox="526 842 1364 873">2. In the Instance Name field, type the name of the resource instance. <div data-bbox="528 921 569 963"></div> <p data-bbox="639 898 1436 993">If a template with the same name exists in the deployment server it will be replaced by the template in the DAA. To prevent this from occurring, repackage the DAA without the resource template.</p> <li data-bbox="486 1010 1106 1041">• New Resource Instance from Existing Template <ol style="list-style-type: none"> <li data-bbox="526 1071 1412 1129">1. In the Template Name drop-down list, choose the template or type the name of template. <li data-bbox="526 1148 1364 1180">2. In the Instance Name field, type the name of the resource instance. <li data-bbox="486 1197 1086 1228">• (HTTP Connector only) New HTTP Connector <ol style="list-style-type: none"> <li data-bbox="526 1257 1460 1316">1. In the HTTP Connector Name field, accept the default name or type a new name. <li data-bbox="526 1335 1420 1394">2. In the HTTP Connector Port field, accept the default port or type a new port. <li data-bbox="486 1413 1476 1472">• From a substitution variable. The substitution variable must resolve to the name of an installed resource instance. <ol style="list-style-type: none"> <li data-bbox="526 1503 1476 1562">1. Check the Map through Substitution Variable checkbox and type the name of a substitution variable in the Substitution Variable field. <p data-bbox="486 1587 1316 1619">Default: The name of the resource instance specified in the composite.</p>

The following actions are added to the script file and object definitions are added to configuration file:


- Create resource instances or application-level substitution variables
- Map application properties to resource instances or variables
- Install resource instances. The `install` action has the `auto-resolve` option.

If you create a resource instance from a resource template in the DAA and the resource templates has properties that reference resource instances, TIBCO Business Studio validates that the DAA has corresponding resource templates and creates and installs resource instances on same nodes as the new

resource instance. If the DAA does not contain corresponding resource templates, TIBCO Business Studio validates that the required resource instances exist on the node. If they do not exist, an error displays.

Wiring Configuration

You can use the Promoted Reference field to configure wires from references that do not have bindings to services in the same environment as the references.

Field	Procedure
Promoted Reference	<p>In the Application/Promoted Service/Binding column of each unresolved reference:</p> <ol style="list-style-type: none"> 1. Click -add wire- in the row containing the unresolved reference. 2. Click . 3. Select a service to wire the reference to. 4. Click the promoted reference.

Running Administrator Command-Line Interface Scripts

After you have configured command-line scripts, you can run them by following the procedures described in the *Workbench User Guide* online help.

Procedure

1. If the script was tokenized when it was generated, optionally update the values in the `DAAName.deployment-config.properties` file.
2. Follow the procedures described in **Help > Help Contents > Workbench User Guide > Tasks > Building Resources > Running Ant build files** .
The specified actions are executed in the Administrator server and a log file named `admincmdling.log` is created in the folder containing the script files.

TIBCO Business Studio Command-Line Interface

The TIBCO Business Studio command-line interface (CLI) provides access to some TIBCO Business Studio functions. In particular, it supports importing projects, creating projects, composites, components, and distributed application archives, regenerating component implementations and libraries, promoting and wiring services and references, and generating distributions.

You can use the CLI for repetitive application of standard actions on large numbers of objects.

The CLI is based on the [Ant](#) open source build tool. TIBCO Business Studio provides Ant tasks to perform TIBCO Business Studio actions. You specify the tasks in a build target within an Ant build file. TIBCO Business Studio supports the following tasks:

- [sds.addProjectDependency](#)
- [sds.createComponent](#)
- [sds.createComposite](#)
- [sds.createDAA](#)
- [sds.createSoaProject](#)
- [sds.generateDeploymentScript](#)
- [sds.generateDistribution](#)
- [tbsimportProjects](#)
- [sds.importProject](#)
- [sds.javait.regenerateComponent](#)
- [sds.javait.regenerateLibraries](#)
- [sds.promoteAll](#)
- [sds.wire](#)

Sample Build Files

Sample build files are provided in *TIBCO_HOME/amx/version/samples/java/helloworld1.zip* and *TIBCO_HOME/amx/version/samples/java/phonebook.zip*.

Running a TIBCO Business Studio Command-Line Interface Script

Procedure

1. Create an Ant build.xml file containing one or more TIBCO Business Studio command-line tasks. For example, the following build file creates an SOA project, a composite in the project, and creates a DAA.

```
<project default="createAll">
  <target name="createAll" depends="createProj, createComposite, createDAA"/>

  <target name="createProj">
    <sds.createSoaProject projectName="aProject"/>
  </target>

  <target name="createComposite">
    <sds.createComposite projectName="aProject" compositeName="aComposite"/>
  </target>

  <target name="createDAA">
    <sds.createDAA projectName="myProject" daaLoc="/MyProject/Deployment Artifacts/
myApp.daa"
      includeApplicationReferences="true" overwriteExistingDAA="true">
```



```

<application compositeName="myAppName">
  <resourceTemplate name="myJDBC" exclude="true"/>
</application>
  <feature id="com.sample.myfeature"/>
</sds.createDAA>
</target>
</project>

```

The following helloworld1 build file imports SOA and Java projects and creates a DAA:

```

<?xml version="1.0" ?>
<project name="helloworld1" basedir="." default="all">

  <property name="projDir" value="C:/helloworld1"/>

  <property name="javaProjectName" value="jv.helloworld1.jv"/>
  <property name="soaProjectName" value="jv.helloworld1.soa"/>

  <property name="importJavaProject" value="${projDir}/${javaProjectName}"/>
  <property name="importSoaProject" value="${projDir}/${soaProjectName}"/>

  <!-- Main target -->
  <target name="all" depends="importProjectsIntoWorkspace,
createApplicationDAA"/>

  <!-- Imports the projects into the workspace -->
  <target name="importProjectsIntoWorkspace">
    <sds.importProject projectLoc="${importJavaProject}" />
    <sds.importProject projectLoc="${importSoaProject}" />
  </target>

  <target name="createApplicationDAA">
    <sds.createDAA
      projectName="${soaProjectName}"
      daaLoc="${soaProjectName}/Deployment Artifacts/${
soaProjectName}.daa"
      includeApplicationReferences="true" overwriteExistingDAA="true">

      <application compositeName="${soaProjectName}"/>
    </sds.createDAA>
  </target>
</project>

```

2. Open a terminal window and change to the folder *TIBCO_HOME/studio/version/eclipse*.
3. Run the **amx_eclipse_ant** command:

- **Windows** **amx_eclipse_ant.exe -buildfile** *path to build.xml* **-data** *workspace* [-vmargs *args*]
- **Linux** **amx_eclipse_ant -buildfile** *path to build.xml* **-data** *workspace* [-vmargs *args*]

Running **amx_eclipse_ant.exe -buildfile "C:/helloworld1/build.xml" -data "C:/hws" -vmargs -Xms256m -Xmx512m** with the preceding helloworld1 build file yields the following:

```

C:\amsg320\studio\<version_number>\eclipse>amx_eclipse_ant.exe -buildfile "C:/
helloworld1/build.xml" -data "C:/hws"
Buildfile: C:/helloworld1/build.xml

importProjectsIntoWorkspace:
[sds.importProject] null
[sds.importProject] ...
...
[sds.importProject] Importing Project 'C:/helloworld1/jv.helloworld1.soa'

createApplicationDAA:
[sds.createDAA] Waited for 47ms for workspace refreshes after building features.

all:

```

```
BUILD SUCCESSFUL
BUILD SUCCESSFUL
Total time: 2 minutes 18 seconds
```

TIBCO Business Studio Command-Line Reference

Option	Required ?	Purpose
-data <i>workspace</i>	Y	The path of the TIBCO Business Studio workspace on which the command-line interface will operate. <i>workspace</i> is the fully-qualified path to a folder in the file system. If the workspace folder does not exist, TIBCO Business Studio creates it.
-vmargs <i>args</i>	N	The Java VM arguments to pass to TIBCO Business Studio. -vmargs must be the last option on the command line. <i>args</i> are VM arguments. Recommended arguments: -Xms256m -Xmx512m

TIBCO Business Studio Command-Line Task Reference



When you use context assist when editing a build file, all elements and attributes appear as lower case.

sds.addProjectDependency

Adds a dependency on a reference project to a source project.

Attribute	Description
referenceProject	The referenced SOA project.
sourceProject	The SOA project to which to add the dependency.

Example

```
<sds.addProjectDependency sourceProject="MySoaProject"
referenceProject="jv.helloworld.soa"/>
```

sds.createComponent

Creates a component from a component implementation. See component development manuals for configuration specific to each component type.

After creating the component and before creating a DAA from the composite containing the component, do the following:

- If you import an implementation project and an SOA project containing a WSDL file, and create the component in another project that references the implementation and SOA projects, add a project dependency from the project containing the component to the implementation and the WSDL projects using the [noPageCitation](#) task.
- If the component implementation has properties, promote the properties using the [sds.promoteAll](#) task.

Attribute or Element	Description
componentName	The name of the component to be created.
compositeName	The name of the composite in which to create the component.
implementationLoc	The path to the implementation file. For component type specific requirements on the path, see the appropriate component development guide.
projectName	The name of the SOA project containing the composite in which to create the component.

Example

```
<sds.createComponent projectName="NewSoaProject"
  compositeName="MyComposite"
  componentName="MyComponent" implementationLoc="MyImpl">
</sds.createComponent>
```

sds.createComposite

Creates a composite in an SOA project.

Attribute	Description
compositeName	The name of the composite to be created.
projectName	The name of the SOA project in which to create the composite.

Example

```
<sds.createComposite projectName="MySoaProject"
  compositeName="MyComposite"/>
```

sds.createDAA

Packages child elements into a distributed application archive.

Attribute or Element	Description
daaLoc	<p>The location of the DAA file to be created. One of:</p> <ul style="list-style-type: none"> • <i>AppName.daa</i> - Creates a DAA with the specified name in the first Deployment Artifacts folder in the <code>projectName</code> project. If no such folder exists the DAA is stored in the root of the <code>projectName</code> project. If the <code>projectName</code> project does not exist then the task fails. • <i>Path/AppName.daa</i> - Creates a DAA in the specified path within the <code>projectName</code> project. Fails if <code>projectName</code> is not specified. Creates any folders that don't exist in the specified path. • <i>/ProjectName/Path/AppName.daa</i> - Creates a DAA in the specified project and path in the workspace. Fails if the project does not exist. Creates any folders that don't exist in the specified path.
failOnUnresolvedReference	Indicated whether the task should fail if any referenced objects cannot be resolved.
id	Optional. An ID for the DAA.
includeApplicationReferences	<p>Optional. When true, all references to resource templates, features, and substitution variable files will be searched for in the scope of each application being packaged. If found and are unique, then the corresponding files will be packaged into the DAA. If not found or are not unique, then the packaging will proceed without packaging the referenced resource templates, features, and substitution variable files.</p> <p>When false, then referenced resource templates, features, and substitution variables files will not be automatically packaged into the DAA. In this case you must explicitly specify the resource templates to package for each application using the appropriate nested child elements.</p> <p>Default: true.</p>
overwriteExistingDAA	<p>Optional. When true, if a DAA already exists in the location specified by the <code>daaLoc</code> attribute this file will be overwritten with the new DAA. When false, if a DAA already exists in the location specified by the <code>daaLoc</code> attribute then the <code>createDAA</code> task fails.</p> <p>Default: true.</p>
projectName	<p>The project name that resolves application elements specified by name. Required if one of the following is true:</p> <ul style="list-style-type: none"> • An application element is referenced by name and does not have a <code>projectName</code> attribute • A fully qualified path is not provided for <code>daaLoc</code>

Attribute or Element	Description
<code>qualifier</code>	Optional. The value to use for the qualifier portion of the version. A qualifier replacement . Default: The qualifier replacement template 'v'yyyy-MM-DD-HHmm.
<code>skipProjectBuild</code>	Indicate that you do not want to build the workspace each time the task is invoked. You can invoke the eclipse.incrementalBuild task to build projects incrementally. Default: false.
<code>version</code>	Optional. The version of the DAA. If not set then no version is assigned.
<i>child elements</i>	One or more instances of: <ul style="list-style-type: none"> • <code>application</code> - Packages a composite or distribution and its referenced composite in the DAA. • <code>feature</code> - Packages a feature.

application Attributes

Attribute or Element	Description
<code>compositeName</code>	The name of the composite to package. This is the name stored in the name field of the composite, not the name of the composite file. In some cases this name will match the filename without the extension, but they may be different. All composite files in <code>Composites</code> folders in the scope of <code>projectName</code> are searched. The scope is the <code>projectName</code> project itself and all referenced projects recursively. If the name is not unique within the scope, the <code>createDAA</code> task will fail. Mutually exclusive with the <code>location</code> attribute.
<code>location</code>	The location in the workspace of a composite or distribution file to package. Mutually exclusive with the <code>compositeName</code> and <code>projectName</code> attributes.
<code>projectName</code>	The name of the project in whose scope the composite specified by the <code>compositeName</code> attribute will be searched for. This is the name of the project as specified in the <code>.project</code> file, not the name of the folder in the filesystem path. In some cases this name will match the folder name, but they may be different. Required when <code>compositeName</code> is specified and <code>projectName</code> is not specified in the <code>createDAA</code> element. Mutually exclusive with the <code>location</code> attribute.

Attribute or Element	Description
resourceTemplate	Packages a resource template.

feature Attributes and Subelements

Attribute or Element	Description
exclude	Optional. Indicate that the specified feature should not be packaged. Use if you want to specify <code>includeApplicationReferences="true"</code> and exclude one or more features. The only valid value for this attribute is true.
id	The ID of the feature to package or exclude. All open projects in the workspace will be searched for a feature project or a custom feature file with a matching ID. If not found or if more than one match is found, the createDAA task will fail.
packagePlugin	Package a pre-built bundle referenced by the feature. To package one bundle you can use either the path element or the location attribute; path and location cannot both used.

packagePlugin Attributes and Subelements


Attribute or Element	Description
location	The fully-qualified path to the bundle.
path	A string containing paths to one or more bundles. You can use ";" or ":" as path separators and Ant will convert it to the platform's local conventions.
updateBundleVersion	Overwrites the version of the bundle. Useful to replace "snapshot" when the bundle was built by Maven.
updateQualifier	Overwrites the qualifier of the bundle. Useful to replace "snapshot" when the bundle was built by Maven. A qualifier replacement.

path Attributes and Subelements

Attribute or Element	Description
id	The name of the Ant property that stores the list of paths specified by the path element. For further information, see resource collections .
dirset	A set of directories.
fileset	A set of files.

Attribute or Element	Description
<code>dir</code>	The fully-qualified path to the root of the <code>dirset</code> set of directories or the <code>fileset</code> set of files.
<code>include</code>	A pattern that specifies the files to include.
<code>exclude</code>	A pattern that specifies the files to exclude.

resourceTemplate Attributes

Attribute	Description
<code>exclude</code>	Optional. A resource template to exclude from being packaged. Use if you want to specify <code>includeApplicationReferences="true"</code> in the <code>createDAA</code> task and then exclude one or more resource templates. The only valid value for this attribute is <code>true</code> .
<code>name</code>	<p>The name of the resource template to package. This is the name stored in the <code>name</code> field of the resource template not the name of the resource template file. In some cases this name will match the filename without the extension, but they may be different.</p> <p>All resource template files in all <code>Resource Templates</code> folders in the scope of <code>projectName</code> will be searched. The scope is the <code>projectName</code> project itself and all referenced projects recursively. If the name is not unique within the scope, the <code>createDAA</code> operation will fail.</p> <div>  <p>You can only list resource templates that are referenced from the composite being packaged. If you list a resource template that is not referenced the <code>createDAA</code> operation will fail.</p> </div>

Example

```
<sds.createDAA projectName="MyProject" daaLoc="/MyProject/Deployment
Artifacts/MyApp.daa"
includeApplicationReferences="false" overwriteExistingDAA="false"
qualifier="20121010" >

  <application compositeName="MyFirstApp"
projectName="MyFirstProject"/>

  <application location="/SomeOtherProject/Composites/
MyApp.composite">
    <resourcetemplate name="MySMTP"/>
    <resourcetemplate name="SomeHTTP"/>
  </application>

  <application location="/MyProject/Composites/MyOtherApp.dist"/>

    <feature id="com.sample.myfeature">
      <packagePlugin updateBundleVersion="2.6.0.qualifier"
updateQualifier="20121010">
        <path id="includePlugin">
          <fileset dir="/ImplJars">
            <include name="*/.jar"/>
          </fileset>
        </path>
      </packagePlugin>
    </feature>
    <feature id="com.sample.myfirstfeature">
  </sds.createDAA>
```

sds.createSoaProject

Creates an empty SOA project.

Attribute	Description
projectName	The name of the SOA project to create.

Example

```
<sds.createSoaProject projectName="MySoaProject"/>
```

sds.generateDeploymentScript

Generate an Administrator CLI script (see [Administrator Command-Line Interface](#)) to deploy a DAA into a remote Administrator server. It generates a deployment script (that is, a build file and a data file) from a DAA that only contains custom features using TIBCO ActiveMatrix Administrator CLI format.

Attribute or Element	Description
adminServer	Optional. The configuration information for the Administrator server.
appConfig	Required. The configuration information for the application created from the DAA. Either <code>singleNodeName</code> or <code>hostAppName</code> is required.

Attribute or Element	Description
<code>createSVars</code>	Optional. Overrides substitution variable values in the generated CLI scripts. To override, enable the generation of <SVar> elements in the <code>deployment-config.xml</code> file, allowing you to manually change the <SVar> values. Default: False.
<code>daaLoc</code>	The absolute path in the workspace to the DAA.
<code>includeCleanupTasks</code>	Optional. If true, the generated script will contain delete and undeploy tasks. Default: false.
<code>outputLoc</code>	The folder to contain the generated script files. Default: The folder containing the DAA.
<code>projectName</code>	The name of the SOA project containing the DAA.
<code>tokenizeScript</code>	Optional. Indicate whether some object attributes, such as the deployment environment and node names, application name and version, and property values, should be externalized into a properties file so that you can optionally update them each time you run the build file. Default: false.
<code>excludeVersionQualifier</code>	Optional. Includes or excludes the application template and feature version qualifiers in the 'Administrator CLI Script Generation' wizard. Default: True Using <code>excludeVersionQualifier=false</code> generates the configuration file with qualifiers. For example, <ul style="list-style-type: none"> • if you set <code>excludeVersionQualifier</code> to true, version is stored as 1.0.0 • If you set <code>excludeVersionQualifier</code> to false, version is stored as 1.0.0.v2014-05-20-1409

adminServer Attributes

Attribute	Description
URL	The URL of the offline server. Default: <code>http://localhost:8120</code> .

Attribute	Description
username	Name of the Administrator user who is executing the task. The user must have the permissions required to execute the actions in the script. Default: root.
password	Password of the Administrator user executing the task. Default: t.

appConfig Attributes

Attribute	Description
appName	The name of the application. Application names cannot contain the characters \, /, :, *, ?, ", <, >, , whitespace, %, #, &, (,), or comma. Default: The name of the DAA.
environmentName	The environment in which to deploy the application. Default: DevEnvironment.
singleNodeName	The node on which to deploy all application fragments.
multipleNodeNames	List of comma-separated Nodes on which to deploy all application fragments.
hostAppName	The name of the application with which to deploy the application.

Example 1

The following example deploys MyApp on the DevNode in the DevEnvironment.

```
<sds.generateDeploymentScript daaLoc="/MyProject/Deployment
Artifacts/MyApp.daa" projectName="MyProject" >
<appconfig appName="MyApp" singleNodeName="DevNode"
environmentName="DevEnvironment" />
</sds.generateDeploymentScript>
```

The following example deploys the sample application on multiple Nodes: DevNodeA, DevNodeB, and DevNodeC.

```
<sds.generateDeploymentScript daaLoc="/MyProject/Deployment
Artifacts/MyApp.daa" projectName="MyProject" >
<appconfig appName="MyApp"
multipleNodeNames="DevNodeA,DevNodeB,DevNodeC"
environmentName="DevEnvironment" />
</sds.generateDeploymentScript>
```

Example 2

```
sds.generateDeploymentScript daaloc="/path/to/MyApplication.daa"
    outputloc="/path/to/generate/deployment/scripts"
    tokenizescript="false"
    excludeVersionQualifier="true" includeCleanupTasks="false">

    <featureconfig environmentName="DevEnvironment" nodeNames="DevNode"
        syncNodes="true" upgrade="true"/>

    <adminserver version="<version_number>" url="http://localhost:
8120"
        username="root" password="test"/>
</sds.generateDeploymentScript>
```

Generating a Deployment Script with Custom Features

You can generate a deployment script from a DAA that only contains custom features. This capability is only available via headless mode (`sds.generateDeploymentScript` target of TIBCO Business Studio CLI). This generates a build file and a data file using TIBCO ActiveMatrix Administrator CLI format.

Sample TIBCO Business Studio CLI script for a deployment script generation:

```
<sds.generateDeploymentScript
daaloc="/path/to/MyApplication.daa"
outputloc="/path/to/generate/deployment/scripts"
tokenizescript="false"
excludeVersionQualifier="true"
includeCleanupTasks="false">
<featureconfig environmentName="DevEnvironment"
nodeNames="DevNode"
syncNodes="true" upgrade="true"/>
<adminserver version="<version_number>" url="http://localhost:8120"
username="root" password="test"/>
</sds.generateDeploymentScript>
```

The generated build file has the following targets:

- `upload.daa`: Uploads the DAA containing features to Administrator.
- `addOrUpgrade.features` or `enable.features`:
 - If `upgrade` is set to `true`, the generated target is called `addOrUpgrade.features`. This target replaces old features on a node with new ones from the DAA. More details are provided below.
 - If `upgrade` is set to `false`, the generated target is called `enable.features`. This target only adds the new features from the DAA but leaves the old features on the node as is.

The `addOrUpgrade` CLI action has been added for the `<Node>` and `<Feature>` elements. The Studio CLI uses this action. This action adds the features from your data file to the nodes. For example, feature `com.acme.myapp.feature 1.5.0` gets added to `DevNode` with status `Marked for Install`. This action also removes any other versions of the same feature, if they were present on the nodes. Features with a different major version are not affected. For example, feature `com.acme.myapp.feature 1.0.0` and `1.6.0` are changed to status `Marked for Uninstall`, since any version not matching "1.5.0" will be marked for removal. However, feature `com.acme.myapp.feature 2.0.0` is left as is, because it has a different major version.

- `reprovision.nodes`: Applies feature changes to runtime in resolve mode, which restarts the nodes to rewire existing applications to use the new versions of features. If `syncNodes` is set to `false`, this target is not generated in the build file.

sds.generateDistribution

Generates a default distribution, with all elements mapped to node `Node1`.

Attribute	Description
compositeName	The name of the composite to distribute.
distributionName	The name of the distribution.
projectName	The name of the SOA project containing the composite in which the component is located.

Example

```
<sds.generateDistribution projectName="jv.helloworld.soa
compositeName="MyComposite"
distributionName="MyDistribution" />
```

tbs.importProjects

Imports one or more Eclipse projects (including ActiveMatrix BPM, SOA, Implementation, Java, and so on) from archives (.zip, .tar, .tar.gz) or project folder structures.





We recommend that you use `tbs.importProjects` rather than `sds.importProject` for TIBCO ActiveMatrix BPM projects, as it provides far more stable post import migration and building.

Two approaches can be adopted when importing projects using this task:

- A simple, attribute-only based, approach suitable for basic scenarios
- A more flexible approach making use of the pre-existing nested Ant tasks, FileSet and DirSet

A single invocation of the task must be configured to import projects from either folder-based structures or archives, but not both.

Attribute	Description
dir	Optional. The root directory to use for simple format folder structure imports.  dir and file are mutually exclusive.
file	Optional. The archive file to use for simple format archive imports (including .zip, .tar, .tar.gz)  dir and file are mutually exclusive.
copyProjects	Optional. Ensures a copy of any imported project is made to the workspace. Applicable only to folder structure imports. Default: true
useArchives	Optional. Ensures archive import is attempted when using nested element format. When a simple format archive import (using the file attribute) is used, the useArchives attribute is not required. It is implicitly set to true. Default: false

Attribute	Description
skipPostImportTask	Optional. Post import tasks are skipped when set to true. Default: false

Archive Import

When importing a single archive from a known location, the simple task format can be used. In this case, the task declaration includes the `file` attribute. The `dir` attribute cannot be used for archive importing.

For example, to import projects from a specific archive (`C:\project_archives\testprojs.zip`), the following simple task declaration can be used:

```
<tbs.importProjects file="C:\project_archives\testprojs.zip"/>
```

For more flexible archive importing, the task's nested format can be used. This permits the importing of multiple archives and a more flexible way to describe the locations of the archives to be imported. To do this, the `useArchives` attribute needs to be set to `true` and one or more of the supported nested elements must be declared to describe candidate archive locations. Both types of nested elements (`FileSet` and `DirSet`) can be used for archive importing.

For example, to import projects from a specific archive (`C:\project_archives\testprojs.zip`), the nested `FileSet` element can be used as follows:

```
<tbs.importProjects useArchives="true"> <fileset  
dir="C:\src\projects" includes="testproj.zip"/>  
</tbs.importProjects>
```

A more useful implementation of the nested element format can be seen in the following example where importing is set up to be selective for `tar` archives:

```
<tbs.importProjects useArchives="true"> <fileset  
dir="C:\src\projects" includes="**/*.tar"/>  
</tbs.importProjects>
```

Project Folder Import

When importing a project or a set of projects contained within a single directory structure, the task's simple format can be used. This, however, does not allow you to be selective about which projects within the directory structure are imported; all identified projects are imported. To use the simple format, the task declaration needs to include the `dir` attribute.

For example, to import all folder-based projects contained within a directory, the following format can be used:

```
<tbs.importProjects dir="C:\folder_projects" />
```

For more flexible folder-based project importing, the task's nested format can be used where one or more `DirSet` elements can be declared to specify the directories to search.

Using the nested format, the previous example could be expressed as:

```
<tbs.importProjects> <dirset dir="C:\src\projects"/>  
</tbs.importProjects>
```

A more useful application of the nested element format could be the following, where only projects contained within directories having a certain substring in their name are considered:

```
<tbs.importProjects> <dirset dir="C:\src\projects">  
<include name="**/*bpmproj"/> </dirset>  
</tbs.importProjects>
```

Optionally, the `copyProjects` attribute can also be used with project folder imports. When set to `false`, no copy of the imported projects is made to the workspace.

sds.importProject

Imports one or more Eclipse projects (including ActiveMatrix BPM, SOA, Implementation, Java, and so on) into a workspace. To import one project you can use either the `path` element or the `projectLoc` attribute; `path` and `projectLoc` cannot both be used.



This function is included for backward compatibility and [tbs.importProjects](#) is recommended for new scripts.

Attribute or Element	Description
<code>copyFiles</code>	Optional. Indicate whether to copy the project files into the workspace. Default: false.
<code>overwrite</code>	Optional. Indicate whether to overwrite a project that already exists in the workspace. If set to false, and the project already exists in the workspace, the task will report an error. Default: false.
<code>path</code>	A string containing paths to one or more projects. You can use ";" or ":" as path separators and Ant will convert it to the platform's local conventions.
<code>projectLoc</code>	The fully-qualified path to the project.

path Attributes and Subelements

Attribute or Element	Description
<code>id</code>	The name of the Ant property that stores the list of paths specified by the <code>path</code> element. For further information, see resource collections .
<code>dirset</code>	A set of directories.
<code>fileset</code>	A set of files.
<code>dir</code>	The fully-qualified path to the root of the <code>dirset</code> set of directories or the <code>fileset</code> set of files.
<code>include</code>	A pattern that specifies the files to include.
<code>exclude</code>	A pattern that specifies the files to exclude.

Example

```
<sds.importProject overwrite="true">  copyFiles="true">
  <path id="projectsToImport">
    <dirset dir="C:/helloworld1/jv.helloworld.soa"/>
    <dirset dir="C:/helloworld1/jv.helloworld.jv"/>
  </path>
</sds.importProject>
```

Example

```
<sds.importProject projectLoc="C:/helloworld1/jv.helloworld.soa" />
```

library Attributes

Attribute	Description
jarworkspacepath	The workspace-qualified path to a library JAR file.
overwriteifexists	Optional. Indicate whether to overwrite a library that already exists in the workspace. If set to false, and the library already exists in the workspace, the task will report an error.
type	The type of data binding technology: <ul style="list-style-type: none"> • Beans • Interface
wdlorschemaworkspacepath	The workspace-qualified path to a WSDL or schema file.

Example

```
<sds.javait.regenerateLibraries>
  <library jarworkspacepath="/Schemas.libs/libs/BaseTypes.xsd.jar"
    overwriteifexists="true" type="Beans"
    wdlorschemaworkspacepath="/Schemas/base/BaseTypes.xsd"/>
  <library jarworkspacepath="/Schemas.libs/libs/
CompanyTypes.xsd.jar"
    overwriteifexists="true" type="Beans"
    wdlorschemaworkspacepath="/Schemas/base/CompanyTypes.xsd"/
>
</sds.javait.regenerateLibraries>
```

sds.promoteAll

Promotes all a component's unwired services and references to the composite level.

Attribute	Description
componentName	The name of the component.
compositeName	The name of the composite containing the component.
projectName	The name of the SOA project containing the composite in which the component is located.

Example

```
<sds.promoteAll projectName="MySoaProject"
compositeName="MyComposite" componentName="MyComponent" />
```

sds.wire

Wires a component reference to a component service.

Attribute	Description
compositeName	The name of the composite containing the service and reference.
projectName	The name of the SOA project containing the composite in which the component is located.
source	The service in the format <i>componentName/serviceName</i> .
target	The reference in the format <i>componentName/referenceName</i> .

Example

```
<sds.wire projectName="MySoaProject" compositeName="MyComposite"  
source="MyComponent/MyService" target="MyComponent/MyReference"/>
```


Logging

The TIBCO ActiveMatrix platform supports application and message flow logging. You configure both types of logging in TIBCO Business Studio run and debug configurations for local nodes and Administrator logging configurations for remote nodes. If the application logging hierarchy is not configured in Administrator, the root of the application logging hierarchy is automatically configured with the same logging configuration as the root of the node logging hierarchy.

Application Logging

The TIBCO ActiveMatrix platform supports application logging to standard out and using a logging API. For simple demonstration applications you can log to standard out. However, for product applications you should use the logging API.

Standard Output and Error Logging

The standard output and error streams from Java programs are directed to the log file of the node on which the program is running. For example, when you run the following example:

```
public class HelloWorldImpl extends AbstractHelloWorldImpl {
    ...
    public HelloResponseDocument sayHello(HelloRequestDocument firstName) {
        ...
        System.out.println("--> Generating Java Hello Component Response...");

        String name = firstName.getHelloRequest()==null||firstName.getHelloRequest().
            equals("")?"Friend":firstName.getHelloRequest();
        HelloResponseDocument resp =
        HelloResponseDocument.Factory.newInstance();
        resp.setHelloResponse("Hi " + name + "! " + "This is the Java component.
        \n");

        System.out.println("--> Java Hello Component Response: \n\t\t" +
            resp.getHelloResponse());
        ...
    }
}
```

the following messages are written to the log:

```
11 Jun 2012 15:51:27,745 [hello1Connector_30] [INFO ] [] stdout - --> Generating
Java Hello Component Response...
11 Jun 2012 15:51:27,808 [hello1Connector_30] [INFO ] [] stdout - --> Java Hello
Component Response:
    Hi Jim! This is the Java component.
```

TIBCO ActiveMatrix Logging

The TIBCO ActiveMatrix platform supports logging using the [Log4J](#). We recommend you use the [Simple Logging Facade for Java](#) package to access log4j. In the following Hello World Java component implementation, the calls to `System.out.println` are replaced with `sl4j` method calls (`logger.info`) that log to a logger named `HelloWorldLogger` at the INFO log level:

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
...
public class HelloWorldImpl extends AbstractHelloWorldImpl {
    ...
    final Logger logger = LoggerFactory.getLogger("HelloWorldLogger");
    ...
    public HelloResponseDocument sayHello(HelloRequestDocument firstName) {
        ...
        logger.info("--> Generating Java Hello Component Response...");

        String name = firstName.getHelloRequest()==null||firstName.getHelloRequest().
            equals("")?"Friend":firstName.getHelloRequest();
        HelloResponseDocument resp = HelloResponseDocument.Factory.newInstance();
```

```

        resp.setHelloResponse("Hi " + name + "! " + "This is the Java component.
\n");

        logger.info("--> Java Hello Component Response: \n\t\t" +
resp.getHelloResponse());
        ...
    }
}

```



By default messages longer than 2K are truncated. To configure the log service to log messages greater than 2K, see Log Service Property Reference in *Administration*.

To log messages on local debug nodes, in TIBCO Business Studio set the Log Level of a run or debug configuration to match the level in the logger method invocations. To specify where log messages are sent on remote nodes, in Administrator create a logging configuration with a logger named HelloWorldLogger for the Hello World application and assign level and appender properties, or accept the default logging configuration which is to use the logging configuration of the node on which the component is deployed.

When you log to HelloWorldLogger at the INFO level and create a run, debug, or logging configuration whose level is set to INFO, the following messages are written to the log:

```

11 Jun 2012 16:57:09,515 [hello1Connector_30] [INFO ] [jv.helloworld1.soa]
HelloWorldLogger - --> Generating Java Hello Component Response...
11 Jun 2012 16:57:09,515 [hello1Connector_30] [INFO ] [jv.helloworld1.soa]
HelloWorldLogger - --> Java Hello Component Response:
        Hi Jim! This is the Java component.

```

Logging to a JMS Queue

Logging information can be sent to and retrieved from JMS queues. To send log entries to a JMS queue, set the appender type of a node or application's logging configuration to JMS Appender.

The following example shows how to retrieve log entries from the queue of a JMS Appender. The log entries consist of key-value pairs of the Base Event Format. For more information on Base Event Format, see *Administration*. For information on the APIs demonstrated in the following example, see *TIBCO Enterprise Message Service User's Guide*.

```

import javax.jms.MapMessage;
...
public class TibjmsAsyncMsgConsumer implements MessageListener {
    ...
    public void start() {
        /* Create the JMS connection. */
        try {
            QueueConnectionFactory factory = new
com.tibco.tibjms.TibjmsQueueConnectionFactory(
            serverUrl);
            connection = factory.createQueueConnection(userName, password);
            session = connection.createQueueSession(false,
javax.jms.Session.AUTO_ACKNOWLEDGE);

            /* Use createQueue() to enable receiving from dynamic queues.*/
            javax.jms.Queue queue = session.createQueue(queueName);
            /* Create a connection, session and consumer */
            queueReceiver = session.createConsumer(queue);
            queueReceiver.setMessageListener(this);
            ...
        }
    }
    public void onMessage(Message message) {
        /* Process message and handle exceptions */
        if (message instanceof MapMessage) {
            try {
                MapMessage mapMessage = (MapMessage) message;
                System.out.println("map message is: " + message.get("_cl.msg"));
                ...
            }
        }
    }
}

```

Payload Data

The following example shows how to save payload data.

```
import java.util.List;
import java.util.UUID;
import javax.activation.MimetypesFileTypeMap;
import org.apache.log4j.Logger;

public class PayloadSample {

    private final static String PAYLOAD_FILE = "./payload.xml";

    public void logPayload() {
        /** Prepare payload data */
        File payloadFile = new File(PAYLOAD_FILE);
        InputStream is = null;
        byte[] payloadBinary = null;
        try {
            is = new FileInputStream(payloadFile);
            payloadBinary = new byte[(int) payloadFile.length()];
            is.read(payloadBinary);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally{
            if (is != null)
                try {
                    is.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
        }

        /** Generate payload key */
        String payloadKey = UUID.randomUUID().toString();

        /** Prepare log record. */
        HashMap<String, Object> logMap = new HashMap<String, Object>();

        logMap.put("_cl.msgId ", "dummyMessageId");
        logMap.put("_cl.msg", "dummy message");

        /** construct payload array which can contain more
         *   than one payload data for one log record. */
        @SuppressWarnings("rawtypes")
        List<Object> payloads = new ArrayList<Object>();

        /** create one payload data. */
        HashMap<String, Object> payloadA = new HashMap<String, Object>();

        /** Set payload key. These are required properties only if the user wants
         *   to know the payload key. If user does not specify this payload key,
         *   logging api will auto-generate one. But user won't be able to get the
         *   auto generated key via Logging API. */
        payloadA.put("_cl.payload.id", payloadKey);

        /** Set payload binary data. These are required properties. */
        payloadA.put("_cl.payload.data", payloadBinary);

        /** Set payload data optional attributes. */

        payloadA.put("_cl.payload.name", PAYLOAD_FILE);
        payloadA.put("_cl.payload.type",
            new MimetypesFileTypeMap().getContentType(payloadFile));
        payloadA.put("_cl.payload.size", payloadFile.length());
        payloadA.put("_cl.payload.MD5", "This is MD5 value of payload binary data.");

    };

    payloadA.put("_cl.payload.TTL", "100");
}
```

```

    /* Add one payload record into list. */
    payloads.add(payloadA);

    /** Put payload key into log record */
    logMap.put("_cl.payloads", payloads);

    /** Log data via Log4j */
    Logger logger = Logger.getLogger(PayloadSample.class);

    if (logger.isInfoEnabled()){
        logger.info(logMap);
    }
}

```

The sample *TIBCO_HOME/amx/version/samples/java/payload.zip* shows how to retrieve and delete payload data.

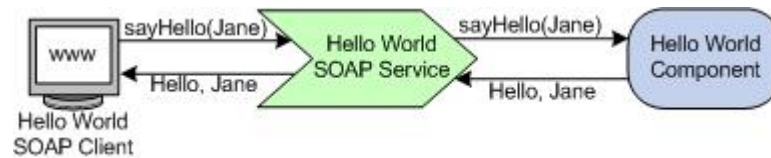
Message Flow Logging

The TIBCO ActiveMatrix platform supports logging the flow of messages between service consumers, services, bindings, components, and references.

In TIBCO Business Studio, you configure message flow logging for a local node by setting the Log Level to DEBUG in a run or debug configuration. In Administrator, the message flow logger *com.tibco.amx.messageflow* controls message flow logging for remote nodes. This logger is based on the node logging hierarchy and thus can be only configured at the node level. When you set a message flow logger level to DEBUG, message flow logging is enabled for all applications deployed on the node.

The following log fragment shows the DEBUG level message log entries for the HelloWorld application depicted in [noPageCitation](#).

HelloWorld Message Flow



```

11 Jun 2012 16:57:09,499 [hello1Connector_30] [DEBUG]
[jv.helloworld1.soa]
com.tibco.amx.messageflow - TIBCO-AMX-BT-SOAP-999100: Request
Message from External Consumer to Promoted Service.
PromotedServiceName=HelloWorldPT/HelloWorldPT,
BindingName=HelloWorld1SOAP, BindingType=SOAP/HTTP, URL=http://sbnb-
t61p.na.tibco.com:9092/helloWorldPT/,
ComponentURI=urn:amx:DevEnvironment/
jv.helloworld1.soa/
JavaHelloComponent_1.0.0.v2012-06-11-1620_inbound_service_HelloWorldP
T/HelloWorldPT_HelloWorld1SOAP,
OperationName=sayHello,
CorrelationId=f4455bdc-3adc-4438-9c68-1128600245b4,
ContextID=f4455bdc-3adc-4438-9c68-1128600245b4,
ParentContextID=null, Message=<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:q0="http://ns.tibco.com/Hello/"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <q0:HelloRequest>Jim</q0:HelloRequest>
  </soapenv:Body>
</soapenv:Envelope>
11 Jun 2012 16:57:09,499 [hello1Connector_30] [DEBUG]
[jv.helloworld1.soa]
com.tibco.amx.messageflow - TIBCO-AMX-CF-999101: Request Message
from TIBCO-BT-SOAP Service Binding to TIBCO-IT-JAVA Component
Service .
SourceComponentURI=urn:amx:DevEnvironment/
jv.helloworld1.soa/
JavaHelloComponent_1.0.0.v2012-06-11-1620_inbound_service_HelloWorldP
T/HelloWorldPT_HelloWorld1SOAP,
DestinationComponentURI=urn:amx:DevEnvironment/
jv.helloworld1.soa/JavaHelloComponent_1.0.0.v2012-06-11-1620,
OperationName={http://ns.tibco.com/Hello/}sayHello,
CorrelationID=f4455bdc-3adc-4438-9c68-1128600245b4,
ContextID=f4455bdc-3adc-4438-9c68-1128600245b4, ParentContextID=null
11 Jun 2012 16:57:09,515 [hello1Connector_30] [DEBUG]
[jv.helloworld1.soa]
com.tibco.amx.messageflow - TIBCO-AMX-CF-999201: Response (Reply)
Message from TIBCO-IT-JAVA Component Service to TIBCO-BT-SOAP
Service Binding .
SourceComponentURI=urn:amx:DevEnvironment/
jv.helloworld1.soa/
JavaHelloComponent_1.0.0.v2012-06-11-1620_inbound_service_HelloWorldP
T/HelloWorldPT_HelloWorld1SOAP,
DestinationComponentURI=urn:amx:DevEnvironment/
jv.helloworld1.soa/JavaHelloComponent_1.0.0.v2012-06-11-1620,
OperationName={http://ns.tibco.com/Hello/}sayHello,
CorrelationID=f4455bdc-3adc-4438-9c68-1128600245b4,
ContextID=6e4780cb-49cc-43f2-b476-ddd1efbe2257,
ParentContextID=f4455bdc-3adc-4438-9c68-1128600245b4
11 Jun 2012 16:57:09,515 [hello1Connector_30] [DEBUG]
[jv.helloworld1.soa]
com.tibco.amx.messageflow - TIBCO-AMX-BT-SOAP-999200: Response
(Reply) Message from Promoted Service to External Consumer.
PromotedServiceName=HelloWorldPT/HelloWorldPT,
BindingName=HelloWorld1SOAP, BindingType=SOAP/HTTP, URL=N/A,
ComponentURI=urn:amx:DevEnvironment/

```

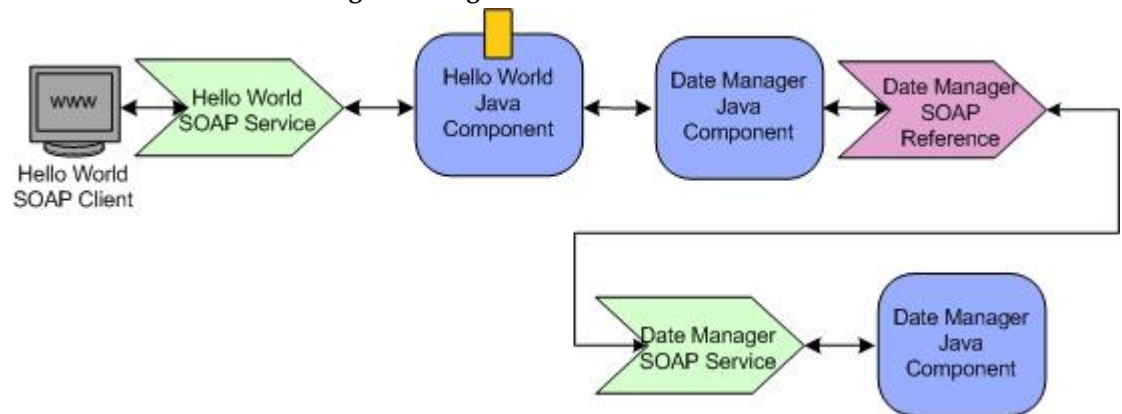
```

jv.helloworld1.soa/
JavaHelloComponent_1.0.0.v2012-06-11-1620_inbound_service_HelloWorldP
T/HelloWorldPT_HelloWorld1SOAP,
OperationName={http://ns.tibco.com/Hello/}sayHello,
CorrelationId=f4455bdc-3adc-4438-9c68-1128600245b4,
ContextID=6e4780cb-49cc-43f2-b476-ddd1efbe2257,
ParentContextID=f4455bdc-3adc-4438-9c68-1128600245b4, Message=<?xml
version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
envelope/"><SOAP-ENV:Header/><SOAP-ENV:Body><HelloResponse
xmlns="http://ns.tibco.com/Hello/">Hi Jim! This is the Java
component.
</HelloResponse></SOAP-ENV:Body></SOAP-ENV:Envelope>

```

The following log fragment shows the DEBUG level message log entries for the Hello World and Date Manager applications depicted in [Hello World and Date Manager Message Flow](#) interspersed with INFO level application logging to stdout.

Hello World and Date Manager Message Flow



```

11 Jun 2012 14:53:12,187 [hello2Connector_30] [DEBUG]
[jv.helloworld2.soa]
com.tibco.amx.messageflow - TIBCO-AMX-BT-SOAP-999100: Request
Message from External Consumer to Promoted Service.
PromotedServiceName=HelloWorldPT/HelloWorldPT,
BindingName=HelloWorld2SOAP, BindingType=SOAP/HTTP,
URL=http://sbnb-t61p.na.tibco.com:9095/helloWorldPT/,
ComponentURI=urn:amx:DevEnvironment/
jv.helloworld2.soa/
JavaHelloWorldComponent_1.0.0.201011151400_inbound_service_HelloWorld
PT/HelloWorldPT_HelloWorld2SOAP,
OperationName=sayHello, CorrelationId=1ab56e2b-
cc6c-4e3a-82ba-22c510ca1239, ContextID=1ab56e2b-
cc6c-4e3a-82ba-22c510ca1239,
ParentContextID=null, Message=<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:q0="http://ns.tibco.com/Hello/"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <q0:HelloRequest>Jim</q0:HelloRequest>
  </soapenv:Body>
</soapenv:Envelope>
11 Jun 2012 14:53:12,187 [hello2Connector_30] [DEBUG]
[jv.helloworld2.soa]
com.tibco.amx.messageflow - TIBCO-AMX-CF-999101: Request Message
from TIBCO-BT-SOAP Service Binding to TIBCO-IT-JAVA Component
Service .
SourceComponentURI=urn:amx:DevEnvironment/
jv.helloworld2.soa/
JavaHelloWorldComponent_1.0.0.201011151400_inbound_service_HelloWorld
PT/HelloWorldPT_HelloWorld2SOAP,
DestinationComponentURI=urn:amx:DevEnvironment/
jv.helloworld2.soa/JavaHelloWorldComponent_1.0.0.201011151400,
OperationName={http://ns.tibco.com/Hello/}sayHello,
CorrelationID=1ab56e2b-cc6c-4e3a-82ba-22c510ca1239,
ContextID=1ab56e2b-cc6c-4e3a-82ba-22c510ca1239, ParentContextID=null
11 Jun 2012 14:53:12,187 [hello2Connector_30] [INFO ]
[jv.helloworld2.soa]
stdout - --> Generating Java Hello Component Response...
11 Jun 2012 14:53:12,187 [hello2Connector_30] [DEBUG]
[jv.helloworld2.soa]
com.tibco.amx.messageflow - TIBCO-AMX-CF-999101: Request Message
from TIBCO-IT-JAVA Component Reference to TIBCO-IT-JAVA Component
Service .
SourceComponentURI=urn:amx:DevEnvironment/
jv.helloworld2.soa/JavaHelloWorldComponent_1.0.0.201011151400,
DestinationComponentURI=urn:amx:DevEnvironment/

```

```

jv.helloworld2.soa/JavaDateManagerComponent_1.0.0.201011151400,
OperationName={http://ns.tibco.com/Date/}getCurrentTime,
CorrelationID=1ab56e2b-cc6c-4e3a-82ba-22c510ca1239,
ContextID=198f1b70-d92a-4351-bb7c-84c86418377e,
ParentContextID=1ab56e2b-cc6c-4e3a-82ba-22c510ca1239
11 Jun 2012 14:53:12,187 [hello2Connector_30] [DEBUG]
[jv.helloworld2.soa]
com.tibco.amx.messageflow - TIBCO-AMX-CF-999101: Request Message
from TIBCO-IT-JAVA Component Reference to TIBCO-BT-SOAP Reference
Binding.
SourceComponentURI=urn:amx:DevEnvironment/
jv.helloworld2.soa/JavaDateManagerComponent_1.0.0.201011151400,
DestinationComponentURI=urn:amx:DevEnvironment/
jv.helloworld2.soa/
JavaDateManagerComponent_1.0.0.201011151400_outbound_reference_DateMa
nagerPT_DateManagerSOAP,
OperationName={http://ns.tibco.com/Date/_gen}getCurrentTime,
CorrelationID=1ab56e2b-cc6c-4e3a-82ba-22c510ca1239,
ContextID=85ee8fe9-3c7f-4731-89df-c1f3ccd00649,
ParentContextID=198f1b70-d92a-4351-bb7c-84c86418377e
11 Jun 2012 14:53:12,203 [hello2Connector_30] [DEBUG]
[jv.helloworld2.soa]
com.tibco.amx.messageflow - TIBCO-AMX-BT-SOAP-999105: Request
Message from Promoted Reference to External Service.
PromotedReferenceName=DateManagerPT, BindingName=DateManagerSOAP,
BindingType=SOAP/HTTP, URL=http://localhost:9097/dateManagerPT/,
ComponentURI=urn:amx:DevEnvironment/
jv.helloworld2.soa/
JavaDateManagerComponent_1.0.0.201011151400_outbound_reference_DateMa
nagerPT_DateManagerSOAP,
OperationName={http://ns.tibco.com/Date/_gen}getCurrentTime,
CorrelationId=1ab56e2b-cc6c-4e3a-82ba-22c510ca1239,
ContextID=85ee8fe9-3c7f-4731-89df-c1f3ccd00649,
ParentContextID=198f1b70-d92a-4351-bb7c-84c86418377e, Message=<?xml
version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
envelope/"><SOAP-ENV:Header/>
<SOAP-ENV:Body><TimeRequest xmlns="http://ns.tibco.com/
Date/">America/Los_Angeles</TimeRequest></SOAP-ENV:Body></SOAP-
ENV:Envelope>
11 Jun 2012 14:53:12,203 [datemgrConnector_26] [DEBUG]
[jv.datemanager.soa]
com.tibco.amx.messageflow - TIBCO-AMX-BT-SOAP-999100: Request
Message from External Consumer to Promoted Service.
PromotedServiceName=DateManagerPT/DateManagerPT,
BindingName=DateManagerSOAP, BindingType=SOAP/HTTP, URL=http://
localhost:9097/dateManagerPT/,
ComponentURI=urn:amx:DevEnvironment/
jv.datemanager.soa/
JavaDateManagerComponent_1.0.0.201005041111_inbound_service_DateManag
erPT/DateManagerPT_DateManagerSOAP,
OperationName=getCurrentTime, CorrelationId=37c1a0b9-4610-4afc-bbf4-
f8966f95a36f, ContextID=37c1a0b9-4610-4afc-bbf4-f8966f95a36f,
ParentContextID=null, Message=<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
envelope/"><SOAP-ENV:Header/>
<SOAP-ENV:Body><TimeRequest xmlns="http://ns.tibco.com/
Date/">America/Los_Angeles</TimeRequest></SOAP-ENV:Body></SOAP-
ENV:Envelope>
11 Jun 2012 14:53:12,203 [datemgrConnector_26] [DEBUG]
[jv.datemanager.soa]
com.tibco.amx.messageflow - TIBCO-AMX-CF-999101: Request Message
from TIBCO-BT-SOAP Service Binding to TIBCO-IT-JAVA Component
Service .
SourceComponentURI=urn:amx:DevEnvironment/
jv.datemanager.soa/
JavaDateManagerComponent_1.0.0.201005041111_inbound_service_DateManag
erPT/DateManagerPT_DateManagerSOAP,
DestinationComponentURI=urn:amx:DevEnvironment/
jv.datemanager.soa/JavaDateManagerComponent_1.0.0.201005041111,
OperationName={http://ns.tibco.com/Date/}getCurrentTime,
CorrelationID=37c1a0b9-4610-4afc-bbf4-f8966f95a36f,

```



```

ContextID=37c1a0b9-4610-4afc-bbf4-f8966f95a36f,
ParentContextID=null
11 Jun 2012 14:53:12,218 [datemgrConnector_26] [INFO ]
[jv.datemanager.soa] stdout - --> Generating Date Manager Component
Response...
11 Jun 2012 14:53:12,218 [datemgrConnector_26] [INFO ]
[jv.datemanager.soa] stdout - --> Date Manager Component Response:
2012-06-11 14:53:12.218
11 Jun 2012 14:53:12,218 [datemgrConnector_26] [DEBUG]
[jv.datemanager.soa]
com.tibco.amx.messageflow - TIBCO-AMX-CF-999201: Response (Reply)
Message from TIBCO-IT-JAVA Component Service to TIBCO-BT-SOAP
Service Binding .
SourceComponentURI=urn:amx:DevEnvironment/
jv.datemanager.soa/
JavaDateManagerComponent_1.0.0.201005041111_inbound_service_DateManag
erPT/DateManagerPT_DateManagerSOAP,
DestinationComponentURI=urn:amx:DevEnvironment/
jv.datemanager.soa/JavaDateManagerComponent_1.0.0.201005041111,
OperationName={http://ns.tibco.com/Date/}getCurrentTime,
CorrelationID=37c1a0b9-4610-4afc-bbf4-f8966f95a36f,
ContextID=680d5ee6-d431-4a82-9f46-b17b7f2a9817,
ParentContextID=37c1a0b9-4610-4afc-bbf4-f8966f95a36f
11 Jun 2012 14:53:12,218 [datemgrConnector_26] [DEBUG]
[jv.datemanager.soa]
com.tibco.amx.messageflow - TIBCO-AMX-BT-SOAP-999200: Response
(Reply) Message from Promoted Service to External Consumer.
PromotedServiceName=DateManagerPT/DateManagerPT,
BindingName=DateManagerSOAP, BindingType=SOAP/HTTP, URL=N/A,
ComponentURI=urn:amx:DevEnvironment/
jv.datemanager.soa/
JavaDateManagerComponent_1.0.0.201005041111_inbound_service_DateManag
erPT/DateManagerPT_DateManagerSOAP,
OperationName={http://ns.tibco.com/Date/}getCurrentTime,
CorrelationId=37c1a0b9-4610-4afc-bbf4-f8966f95a36f,
ContextID=680d5ee6-d431-4a82-9f46-b17b7f2a9817,
ParentContextID=37c1a0b9-4610-4afc-bbf4-f8966f95a36f, Message=<?xml
version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
envelope/"><SOAP-ENV:Header/>
<SOAP-ENV:Body><TimeResponse xmlns="http://ns.tibco.com/
Date/">2012-06-11 14:53:12.218</TimeResponse></SOAP-ENV:Body></SOAP-
ENV:Envelope>
11 Jun 2012 14:53:12,218 [hello2Connector_30] [DEBUG]
[jv.helloworld2.soa]
com.tibco.amx.messageflow - TIBCO-AMX-BT-SOAP-999205: Response
(Reply) Message from External Service to Promoted Reference.
PromotedReferenceName=DateManagerPT, BindingName=DateManagerSOAP,
BindingType=SOAP/HTTP, URL=N/A,
ComponentURI=urn:amx:DevEnvironment/
jv.helloworld2.soa/
JavaDateManagerComponent_1.0.0.201011151400_outbound_reference_DateMa
nagerPT_DateManagerSOAP,
OperationName=getCurrentTime, CorrelationId=1ab56e2b-
cc6c-4e3a-82ba-22c510ca1239, ContextID=0bd769d1-26cc-430a-
bdf2-52849cad9a27,
ParentContextID=85ee8fe9-3c7f-4731-89df-c1f3ccd00649, Message=<?xml
version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
envelope/"><SOAP-ENV:Header/>
<SOAP-ENV:Body><TimeResponse xmlns="http://ns.tibco.com/
Date/">2012-06-11 14:53:12.218</TimeResponse></SOAP-ENV:Body></SOAP-
ENV:Envelope>
11 Jun 2012 14:53:12,234 [hello2Connector_30] [DEBUG]
[jv.helloworld2.soa]
com.tibco.amx.messageflow - TIBCO-AMX-CF-999201: Response (Reply)
Message from TIBCO-BT-SOAP Reference Binding to TIBCO-IT-JAVA
Component Reference.
SourceComponentURI=urn:amx:DevEnvironment/
jv.helloworld2.soa/JavaDateManagerComponent_1.0.0.201011151400,
DestinationComponentURI=urn:amx:DevEnvironment/
jv.helloworld2.soa/

```

```

JavaDateManagerComponent_1.0.0.201011151400_outbound_reference_DateMa
nagerPT_DateManagerSOAP,
OperationName={http://ns.tibco.com/Date/_gen}getCurrentTime,
CorrelationID=1ab56e2b-cc6c-4e3a-82ba-22c510ca1239,
ContextID=0bd769d1-26cc-430a-bdf2-52849cad9a27,
ParentContextID=85ee8fe9-3c7f-4731-89df-clf3ccd00649
11 Jun 2012 14:53:12,234 [hello2Connector_30] [INFO ]
[jv.helloworld2.soa] stdout - --> Date Manager Component Response:
2012-06-11 14:53:12.218
11 Jun 2012 14:53:12,234 [hello2Connector_30] [DEBUG]
[jv.helloworld2.soa]
com.tibco.amx.messageflow - TIBCO-AMX-CF-999201: Response (Reply)
Message from TIBCO-IT-JAVA Component Service to TIBCO-IT-JAVA
Component Reference.
SourceComponentURI=urn:amx:DevEnvironment/
jv.helloworld2.soa/JavaHelloWorldComponent_1.0.0.201011151400,
DestinationComponentURI=urn:amx:DevEnvironment/
jv.helloworld2.soa/JavaDateManagerComponent_1.0.0.201011151400,
OperationName={http://ns.tibco.com/Date/}getCurrentTime,
CorrelationID=1ab56e2b-cc6c-4e3a-82ba-22c510ca1239,
ContextID= 7cc827f9-6ab6-4048-9023-f86f2ed089c5,
ParentContextID=198f1b70-d92a-4351-bb7c-84c86418377e
11 Jun 2012 14:53:12,234 [hello2Connector_30] [INFO ]
[jv.helloworld2.soa] stdout - --> Java Hello Component Response:
    Hi Jim! This is the Java component.
The current time is 2012-06-11 14:53:12.218.
11 Jun 2012 14:53:12,234 [hello2Connector_30] [DEBUG]
[jv.helloworld2.soa]
com.tibco.amx.messageflow - TIBCO-AMX-CF-999201: Response (Reply)
Message from TIBCO-IT-JAVA Component Service to TIBCO-BT-SOAP
Service Binding .
SourceComponentURI=urn:amx:DevEnvironment/
jv.helloworld2.soa/
JavaHelloWorldComponent_1.0.0.201011151400_inbound_service_HelloWorld
PT/HelloWorldPT_HelloWorld2SOAP,
DestinationComponentURI=urn:amx:DevEnvironment/
jv.helloworld2.soa/JavaHelloWorldComponent_1.0.0.201011151400,
OperationName={http://ns.tibco.com/Hello/}sayHello,
CorrelationID=1ab56e2b-cc6c-4e3a-82ba-22c510ca1239,
ContextID=8cbca586-fce6-43d3-ac36-02479eea07d9,
ParentContextID=1ab56e2b-cc6c-4e3a-82ba-22c510ca1239
11 Jun 2012 14:53:12,234 [hello2Connector_30] [DEBUG]
[jv.helloworld2.soa]
com.tibco.amx.messageflow - TIBCO-AMX-BT-SOAP-999200: Response
(Reply) Message from Promoted Service to External Consumer.
PromotedServiceName=HelloWorldPT/HelloWorldPT,
BindingName=HelloWorld2SOAP, BindingType=SOAP/HTTP, URL=N/A,
ComponentURI=urn:amx:DevEnvironment/
jv.helloworld2.soa/
JavaHelloWorldComponent_1.0.0.201011151400_inbound_service_HelloWorld
PT/HelloWorldPT_HelloWorld2SOAP,
OperationName={http://ns.tibco.com/Hello/}sayHello,
CorrelationId=1ab56e2b-cc6c-4e3a-82ba-22c510ca1239,
ContextID=8cbca586-fce6-43d3-ac36-02479eea07d9,
ParentContextID=1ab56e2b-cc6c-4e3a-82ba-22c510ca1239, Message=<?xml
version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
envelope/"><SOAP-ENV:Header/>
<SOAP-ENV:Body><HelloResponse xmlns="http://ns.tibco.com/Hello/">Hi
Jim! This is the Java component.
The current time is 2012-06-11 14:53:12.218.</HelloResponse></SOAP-
ENV:Body></SOAP-ENV:Envelope>

```

Logging Payload Reference

Payload Field Keys

Field Key	Description
<code>_cl.payload.id</code>	Payload ID.
<code>_cl.payload.name</code>	Auto-generated file name.
<code>_cl.payload.type</code>	Auto-detected mime type of the payload file.
<code>_cl.payload.uri</code>	URI of payload data. This key has been deprecated.
<code>_cl.payload.size</code>	File size of payload file.
<code>_cl.payload.MD5</code>	MD5 value of payload file.
<code>_cl.payload.TTL</code>	Time to leave of payload data. The payload data will be automatically purged when it reaches the time to leave. The unit of TTL is hour.
<code>_cl.payload.data</code>	Binary data of payload file.

IPv6 Support

If an object has a property that can contain an IP address, the address is usually set to the unspecified IP address (0.0.0.0). That means the object listens on IPv4 and IPv6 addresses. By default clients use the IPv4 address. You can override this behavior so that clients use the IPv6 address.

Prerequisites

Before using an IPv6 supported network, perform the following tasks:

1. Complete all the network configuration changes required for network traffic routing.
2. Enable all physical machines participating in the installation topology for IPv4 and IPv6 addressing in dual-stack IP implementations.
3. Configure the names of all machines to resolve to at least one IPv4 or IPv6 address.
4. Configure clients to communicate with the servers in one of the following ways:
 - a. Use explicit IPv4 or IPv6 addresses.
 - b. Use the addresses returned by the address translation mechanism (DNS or local host files) performed on the machine name.

IPv6 Address Support

IPv6 addresses are supported by machine names and URLs in the following tools and objects:

- TIBCO Configuration Tool
- Administrator and TIBCO Business Studio wizards and CLI property files
- Components that use dynamic wiring
- Resource templates

IPv6 Address Representation

IPv6 address representation is described in the [IPv6 Addressing Architecture](#) and [Format for Literal IPv6 Addressing in URLs](#) specifications, and summarized in [IPv6 Address Representation](#).

IPv6 Address Representation

Address Type	Representation
All	<p>Eight fields of four hexadecimal digits, where each field is separated by a colon. If the field is non-zero there must be at least one digit. For example, 2001:db8:1234:ffff:4354:45ab:3455:ab45. You can apply the following shortening procedures:</p> <ul style="list-style-type: none"> • Omit leading zeros in a field. For example, :00db: can be represented as :db:. • Replace one or more consecutive fields of zeros and separators (: 0:0:0:0:) with a single empty field (::). For example, 2001:db8:0:0:0:0:3455:ab45 can be represented as 2001:db8::3455:ab45.
Localhost or loopback	0:0:0:0:0:0:0:1 or ::1.

Address Type	Representation
Unspecified	0:0:0:0:0:0:0 or ::. This address is equivalent to the unspecified IPv4 address 0.0.0.0.
Embedded in a URL	Enclose the address in square brackets ([]). For example, the URL of an Administrator server running on a machine at the address FEDC:BA98:7654:3210:FEDC:BA98:7654:3210 is http://[FEDC:BA98:7654:3210:FEDC:BA98:7654:3210]:8120/amxadministrator.

IP Address Use and Resolution

The default configuration of the Administrator server network adapter is the unspecified IP address (0.0.0.0), which means that it listens on IPv4 and IPv6 addresses. When clients access the Administrator server by machine name, the name lookup resolves to both addresses. By default, Administrator clients use the IPv4 address. To override this behavior and use the IPv6 address, set the value of the JVM system property `java.net.preferIPv6Addresses` to `true` in the Administrator Node. The Managing Nodes section in *Administration Guide* explains how to set a JVM property for a Node.