TIBCO ActiveMatrix® Adapter Service Engine for IBM i

Configuration and Deployment

Software Release 6.0
March 2010

TIBCO®
The Power of Now®

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 LICENSE.PDF) 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.

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

TIB, TIBCO, TIBCO Adapter, TIBCO ActiveMatrix, Predictive Business, Information Bus, The Power of Now, TIBCO ActiveMatrix BusinessWorks, TIBCO Rendezvous, TIBCO Administrator, TIBCO Designer, TIBCO Runtime Agent, TIBCO Hawk, TIBCO Enterprise Message Service, TIBCO Designer Add-in for TIBCO Business Studio, TIBCO ActiveMatrix Service Grid, TIBCO ActiveMatrix Service Bus, TIBCO ActiveMatrix BusinessWorks Service Engine, TIBCO ActiveEnterprise, and TIBCO Business Studio, are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

EJB, Java EE, J2EE, and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

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

THIS SOFTWARE MAY BE AVAILABLE ON MULTIPLE OPERATING SYSTEMS. HOWEVER, NOT ALL OPERATING SYSTEM PLATFORMS FOR A SPECIFIC SOFTWARE VERSION ARE RELEASED AT THE SAME TIME. SEE THE README.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.

Copyright © 2008-2010 TIBCO Software Inc. ALL RIGHTS RESERVED.

TIBCO Software Inc. Confidential Information

Contents

Figures vii

Tables ix

Preface xi

Related Documentation xii
TIBCO ActiveMatrix Adapter Service Engine for IBM i Documentation xii
Other TIBCO Product Documentation xiii
Third-Party Documentation. xiii

Typographical Conventions xiv

How to Contact TIBCO Support. xvii

Chapter 1 Introduction. 1

Overview 2

Chapter 2 Working with TIBCO Business Studio 3

Starting TIBCO Business Studio 4

Creating Adapter Projects 6

Importing TIBCO Designer Projects 7
Differences in Imported Projects 8

Importing Projects into Workspace 9

Chapter 3 Preparing IBM System i 11

Loading a Library on an IBM System i Machine 12

Adding the Library to a Library List 13

Starting Host Server Daemons 14

Chapter 4 Getting Started 15

Overview 16

Prerequisites 17

Configuring the Adapter Service Engine 18
Configuration Task Sequence. 18

Configuring an ActiveMatrix SOA Project 22

Running the Adapter Service Engine 23

Chapter 5 Configuring Adapter Instance 25

IBM i Connection Configuration Shared Resource 26
Creating an IBM i Connection 26
Configuring an IBM i Connection 27

IBM i Business Object 28
Importing IBM i Business Object 28
Editing IBM i Business Objects 31
Converting Business Objects into AESchemas 33

IBM i Adapter Instance Configuration 35
Configuration Tab 35
Adapter Options Tab 37
Adapter Services Tab 39
Logging Sinks Tab 49

Chapter 6 Configuring ActiveMatrix SOA Project 53

Creating an ActiveMatrix SOA Project 54

Generating WSDL Files 55

Configuring Composites 56

Creating Service Assembly and Service Assembly Archive 60

Chapter 7 Deploying the Service Assembly 61

Overview 62

Using the EAR2SA Tool 63

Deploying a Service Assembly Archive 64
Uploading a Service Assembly Archive 64
Configuring the Service Assembly 64
Deploying the Service Assembly 65
Starting the Service Assembly 66

Chapter 8 Advanced Topics 67

Managing Logging 68

Using the Adapter Service Engine with a Revision Control System 69

Using Substitution Variables 70
Variable Specification 70
Predefined Substitution Variables 72

Chapter 9 Monitoring the Adapter Using TIBCO Hawk 75

Overview 76

Starting TIBCO Hawk Software 77
The Auto-Discovery Process 78
Invoking Microagent Methods 79
Available Microagents 82
activateTraceRole() 85
deactivateTraceRole() 86
getActivityStatistics() 87
getActivityStatisticsByOperation() 88
getActivityStatisticsByService() 89
getAdapterServiceInformation() 90
getComponents() 91
getConfig() 92
getConfigProperties() 93
getConnectionStatistics() 94
getHostInformation() 95
getQueueStatistics() 96
getRvConfig() 97
getStatus() 98
getThreadStatistics() 99
getTraceSinks() 100
getVersion() 101
_onUnsolictedMsg() 102
preRegisterListener() 103
resetActivityStatistics() 104
resetConnectionStatistics() 105
resetThreadStatistics() 106
reviewLedger() 107
setTraceSinks() 109
stopApplicationInstance() 110
unRegisterListener() 111

Appendix A Status Messages 113

Overview 114
Trace Message Fields 116
Status Messages 118

Index 127

Figures

Figure 1 TIBCO ActiveMatrix Product Family 2
Figure 2 TIBCO Business Studio Workbench Window 5
Figure 3 Create an Adapter Project 6
Figure 4 Import TIBCO Designer Project 7
Figure 5 Import Projects Created in Another Workspace 9
Figure 6 Select Asset Types 19
Figure 7 Create an IBM i Connection 26
Figure 8 IBM i Connection Shared Resource Editor 27
Figure 9 Import IBM i Business Object 28
Figure 10 IBM i Business Object Import Wizard 29
Figure 11 Select from the Application Tree 31
Figure 12 IBM i Adapter Business Object Editor 32
Figure 13 Select BO 33
Figure 14 Schema Editor 34
Figure 15 Log Sinks 49
Figure 16 Create an ActiveMatrix SOA Project 54
Figure 17 Add Resources to a Component 57
Figure 18 Component and Composite Service Binding 59
Figure 19 General Tab of Composite Service 59
Figure 20 Add Files to the Revision Control System 69
Figure 21 Substitution Variables 71
Figure 22 Substitution Variables in TIBCO ActiveMatrix Administrator 72
Figure 23 TIBCO Hawk Enterprise Monitor 78
Figure 24 Microagents, Methods and Arguments Dialog 79
Figure 25 Select Microagent and Method 80
Figure 26 Invocation Result Dialog 81
Figure 27 Trace Message 114

Tables

Table 1 General Typographical Conventions . . . xiv
Table 2 Syntax Typographical Conventions . . . xv
Table 3 IBM i Connection Configuration Parameters . . . 27
Table 4 Field Description of Configuration Tab . . . 35
Table 5 Fields of Adapter Options Tab . . . 37
Table 6 Publisher Options . . . 39
Table 7 Subscriber Options . . . 41
Table 8 RPC Server Options . . . 43
Table 9 RPC Client Options . . . 47
Table 10 Field Description of Log Sinks Configuration . . . 50
Table 11 Fields of the General Tab . . . 58
Table 12 Properties for Common Logging Feature . . . 68
Table 13 Predefined Substitution Variables . . . 72
Table 14 Microagent Methods . . . 82
Table 15 Input parameters of activateTraceRole() . . . 85
Table 16 Input parameters of deactivateTraceRole() . . . 86
Table 17 Input parameter of getActivityStatistics() . . . 87
Table 18 Returns of getActivityStatistics() . . . 87
Table 19 Input parameter of getActivityStatisticsByOperation() . . . 88
Table 20 Returns of getActivityStatisticsByOperation() . . . 88
Table 21 Input parameter of getActivityStatisticsByService() . . . 89
Table 22 Returns of getActivityStatisticsByService() . . . 89
Table 23 Input parameter of getAdapterServiceInformation() . . . 90
Table 24 Returns of getAdapterServiceInformation() . . . 90
Table 25 Input parameters of getComponents() . . . 91
Table 26 Returns of getComponents() . . . 91
Table 27 Returns of getConfig() . . . 92
Table 28 Input parameter of getConfigProperties() . . . 93

Table 29 Returns of getConfigProperties() 93
Table 30 Returns of getConnectionStatistics() 94
Table 31 Returns of getHostInformation() 95
Table 32 Returns of getQueueStatistics() 96
Table 33 Input parameter of getRvConfig() 97
Table 34 Returns of getRvConfig() 97
Table 35 Returns of getStatus() 98
Table 36 Returns of getThreadStatistics() 99
Table 37 Input parameters of getTraceSinks() 100
Table 38 Returns of getTraceSinks() 100
Table 39 Returns of getVersion() 101
Table 40 Input parameters of preRegisterListener() 103
Table 41 Input parameters of reviewLedger() 107
Table 42 Returns of reviewLedger() 107
Table 43 Input parameters of setTraceSinks() 109
Table 44 Input parameters of unRegisterListener() 111
Table 45 Tracing Fields 116
Table 46 Status Messages 118

Preface

TIBCO ActiveMatrix Adapter Service Engine for IBM i allows one-way (publish or subscribe) or two-way (request-response) message exchange between TIBCO environment and applications running on IBM System i machines.

This manual explains the configuration and deployment features of the adapter service engine.

Topics

- Related Documentation, page xii
- Typographical Conventions, page xiv
- How to Contact TIBCO Support, page xvii

Related Documentation

This section lists documentation resources you may find useful.

TIBCO ActiveMatrix Adapter Service Engine for IBM i Documentation

The following documents form the TIBCO ActiveMatrix Adapter Service Engine for IBM i documentation set:

- *TIBCO ActiveMatrix Adapter Service Engine for IBM i Installation* Read this manual for instructions on site preparation and installation.
- *TIBCO ActiveMatrix Adapter Service Engine for IBM i Configuration and Deployment* Read this manual for instructions on creating and configuring adapter projects. Instructions on how to deploy the projects are also included.
- *TIBCO ActiveMatrix Adapter Service Engine for IBM i Examples* Read this manual to work through the examples provided with the adapter service engine.
- *TIBCO ActiveMatrix Adapter Service Engine for IBM i Release Notes* Read the release notes for a list of new and changed features. This document also contains lists of closed and known issues for this release.

Before TIBCO ActiveMatrix Adapter Service Engine for IBM i can be installed, you must first install the TIBCO ActiveMatrix Adapter for IBM i. The following documents form the standalone adapter documentation set:

- *TIBCO ActiveMatrix Adapter for IBM i Concepts* Read this manual before reading any other book in the documentation set to familiarize yourself with the product and its uses.
- *TIBCO ActiveMatrix Adapter for IBM i Installation* Read this manual for instructions on site preparation and installation.
- *TIBCO ActiveMatrix Adapter for IBM i Configuration and Deployment* Read this manual for instructions on creating and configuring standalone adapter projects. Information on deploying adapter projects is also included.
- *TIBCO ActiveMatrix Adapter for IBM i Examples* Read this manual to work through the examples provided with the adapter.
- *TIBCO ActiveMatrix Adapter for IBM i Release Notes* Read the release notes for a list of new and changed features. This document also contains lists of closed and known issues for this release.

Other TIBCO Product Documentation

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

- TIBCO ActiveMatrix® Administrator
- TIBCO ActiveMatrix BusinessWorks™
- TIBCO ActiveMatrix BusinessWorks™ Service Engine
- TIBCO ActiveMatrix® Service Grid
- TIBCO ActiveMatrix® Service Bus
- TIBCO Business Studio™
- TIBCO Enterprise Message Service™
- TIBCO Hawk®
- TIBCO Rendezvous®
- TIBCO Runtime Agent™

Third-Party Documentation

You may also find it useful to read the IBM documentation on the following web sites:

- http://www-03.ibm.com/systems/i/software/toolbox/
- http://jt400.sourceforge.net/

Typographical Conventions

The following typographical conventions are used in this manual.

Table 1 General Typographical Conventions

Convention	Use
TIBCO_HOME *ENV_HOME* *AMX_HOME*	Many TIBCO products must be installed within the same home directory. This directory is referenced in documentation as *TIBCO_HOME*. The value of *TIBCO_HOME* depends on the operating system. For example, on Windows systems, the default value is `C:\tibco`. Other TIBCO products are installed into an installation environment. Incompatible products and multiple instances of the same product are installed into different installation environments. The directory into which such products are installed is referenced in documentation as *ENV_HOME*. The value of *ENV_HOME* depends on the operating system. For example, on Windows systems the default value is `C:\tibco`. TIBCO ActiveMatrix installs into a directory within *TIBCO_HOME*. This directory is referenced in documentation as *AMX_HOME*. The value of *AMX_HOME* depends on the operating system. For example on Windows systems, the default value is `C:\tibco\amx\`. TIBCO ActiveMatrix Adapter Service Engine for IBM i installs in a directory within *AMX_HOME*.
`code font`	Code font identifies commands, code examples, filenames, pathnames, and output displayed in a command window. For example: Use `MyCommand` to start the foo process.
`bold code font`	Bold code font is used in the following ways: • In procedures, to indicate what a user types. For example: Type **`admin`**. • In large code samples, to indicate the parts of the sample that are of particular interest. • In command syntax, to indicate the default parameter for a command. For example, if no parameter is specified, `MyCommand` is enabled: `MyCommand [`**`enable`**` | disable]`

Table 1 General Typographical Conventions (Cont'd)

Convention	Use
italic font	Italic font is used in the following ways: • To indicate a document title. For example: See *TIBCO ActiveMatrix BusinessWorks Concepts*. • To introduce new terms For example: A portal page may contain several portlets. *Portlets* are mini-applications that run in a portal. • To indicate a variable in a command or code syntax that you must replace. For example: `MyCommand` *PathName*
Key combinations	Key name separated by a plus sign indicate keys pressed simultaneously. For example: Ctrl+C. Key names separated by a comma and space indicate keys pressed one after the other. For example: Esc, Ctrl+Q.
	The note icon indicates information that is of special interest or importance, for example, an additional action required only in certain circumstances.
	The tip icon indicates an idea that could be useful, for example, a way to apply the information provided in the current section to achieve a specific result.
	The warning icon indicates the potential for a damaging situation, for example, data loss or corruption if certain steps are taken or not taken.

Table 2 Syntax Typographical Conventions

Convention	Use
`[ ]`	An optional item in a command or code syntax. For example: `MyCommand [optional_parameter] required_parameter`
`|`	A logical `OR` that separates multiple items of which only one may be chosen. For example, you can select only one of the following parameters: `MyCommand para1 | param2 | param3`

Table 2 Syntax Typographical Conventions (Cont'd)

Convention	Use

`{ }`	A logical group of items in a command. Other syntax notations may appear within each logical group.
	
	For example, the following command requires two parameters, which can be either the pair `param1` and `param2`, or the pair `param3` and `param4`.
	
	`MyCommand {param1 param2} | {param3 param4}`
	
	In the next example, the command requires two parameters. The first parameter can be either `param1` or `param2` and the second can be either `param3` or `param4`:
	
	`MyCommand {param1 | param2} {param3 | param4}`
	
	In the next example, the command can accept either two or three parameters. The first parameter must be `param1`. You can optionally include `param2` as the second parameter. And the last parameter is either `param3` or `param4`.
	
	`MyCommand param1 [param2] {param3 | param4}`

How to Contact TIBCO Support

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

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

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

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

  https://support.tibco.com

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

Chapter 1 Introduction

This chapter provides an introduction to TIBCO ActiveMatrix Adapter Service Engine for IBM i (adapter service engine).

Topics

- Overview, page 2

Overview

TIBCO ActiveMatrix Adapter Service Engine for IBM i is a gateway for TIBCO ActiveMatrix Adapter for IBM i to the Service Oriented Architecture (SOA) world. The TIBCO ActiveMatrix Adapter Service Engine for IBM i provides an ActiveMatrix container to deploy adapter projects using the TIBCO ActiveMatrix Administrator.

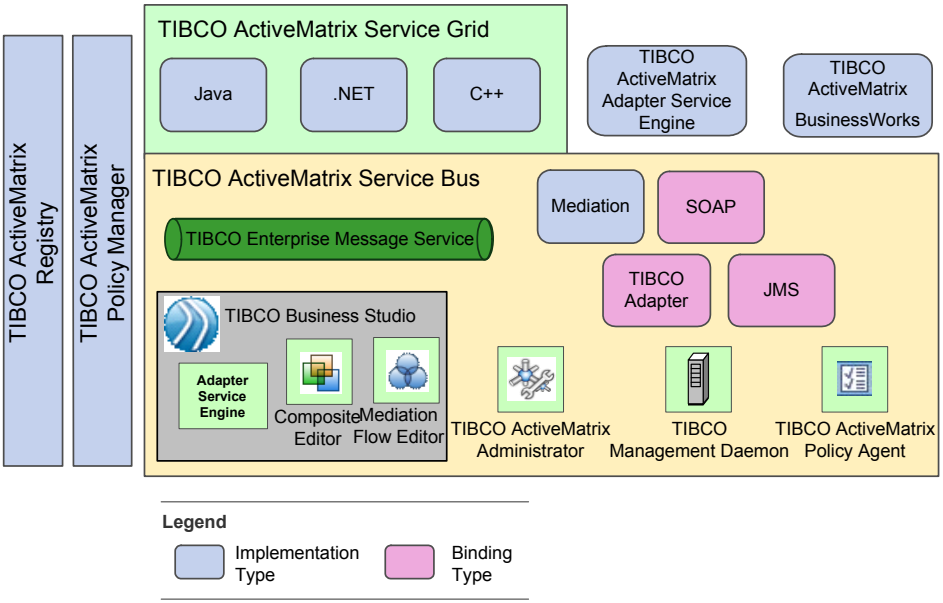
This manual describes the process of creating and configuring an adapter instance, packaging it into a service assembly and then deploying the services.

The design time tool is TIBCO Business Studio and the services are deployed using TIBCO ActiveMatrix Administrator.

Before proceeding, you should be familiar with the concepts and the terminology used by TIBCO ActiveMatrix Adapter Service Engine for IBM i and TIBCO ActiveMatrix platform.

Figure 1 shows the TIBCO ActiveMatrix product family.

Figure 1 TIBCO ActiveMatrix Product Family



Chapter 2 Working with TIBCO Business Studio

The TIBCO ActiveMatrix development tools consist of TIBCO Business Studio Workbench and a set of TIBCO ActiveMatrix plug-ins. This chapter describes how to start TIBCO Business Studio, create a new adapter project, and import projects and files into the project.

For information on TIBCO Business Studio, refer to the *Workbench User Guide* in the Workbench online help. To view the online help, select **Help > Help Contents**.

Topics

- Starting TIBCO Business Studio, page 4
- Creating Adapter Projects, page 6
- Importing TIBCO Designer Projects, page 7
- Importing Projects into Workspace, page 9

Starting TIBCO Business Studio

When you start TIBCO Business Studio for the first time after installing TIBCO ActiveMatrix Adapter Service Engine for IBM i, ensure that you use the `-clean` option.

To start TIBCO Business Studio:

1. Execute one of the following platform-specific command:
   - On Microsoft Windows:

     select **Start > All Programs > TIBCO >** *AMX_HOME* **> TIBCO Business Studio 2.2 > TIBCO Business Studio**

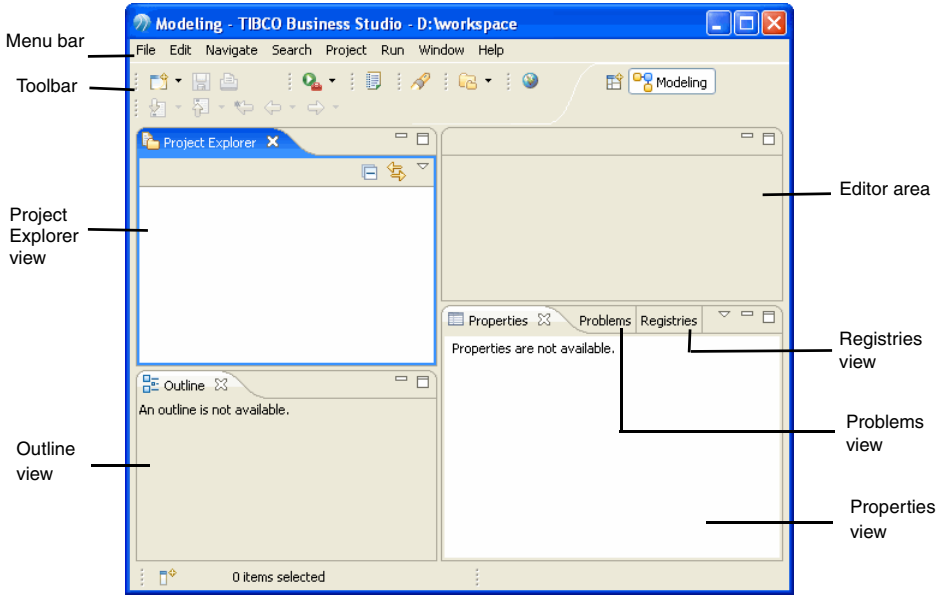
     or

     run the executable from the command line

     AMX_HOME`\BusinessStudio\TIBCOBusinessStudio.exe`
   - On Unix:

     run *AMX_HOME*`/BusinessStudio/TIBCOBusinessStudio`
2. In the Workspace Launcher, accept the default workspace or browse to or create a new workspace. Click **OK**.
3. The first time you run TIBCO Business Studio, it displays a Welcome screen. Dismiss the screen by clicking the **Close** button ☒ in the title bar.

Figure 2 TIBCO Business Studio Workbench Window



Creating Adapter Projects

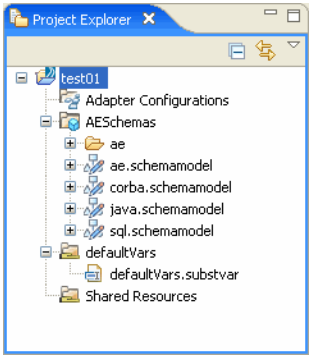
Adapter projects contain adapter service engine configurations, related shared resources and business functionality that needs to be enabled for the functioning of the adapter service engines. Adapter projects are created using TIBCO Business Studio. To begin the configuration process, you must first create a project for the adapter.

1. Start TIBCO Business Studio.
2. From the File menu, select **New** > **Project**.
3. In the New Project dialog, expand **TIBCO ActiveMatrix Adapters**, select **Adapter Project** as the wizard, and then click **Next**.
4. Type *AdapterProjectName* in the Project name field.
5. Specify names for folders.
   - To specify custom names for Adapter Configurations and Shared Resources folders, click **Next**.
   - To accept default values, click **Finish**.

A new project is created with the following folders: `Adapter Configurations`, `AESchemas`, `defaultVars`, and `Shared Resources`.

Figure 3 shows the structure of a project created using default values.

Figure 3 Create an Adapter Project



Global variables in TIBCO Designer are referred to as substitution variables in TIBCO Business Studio.

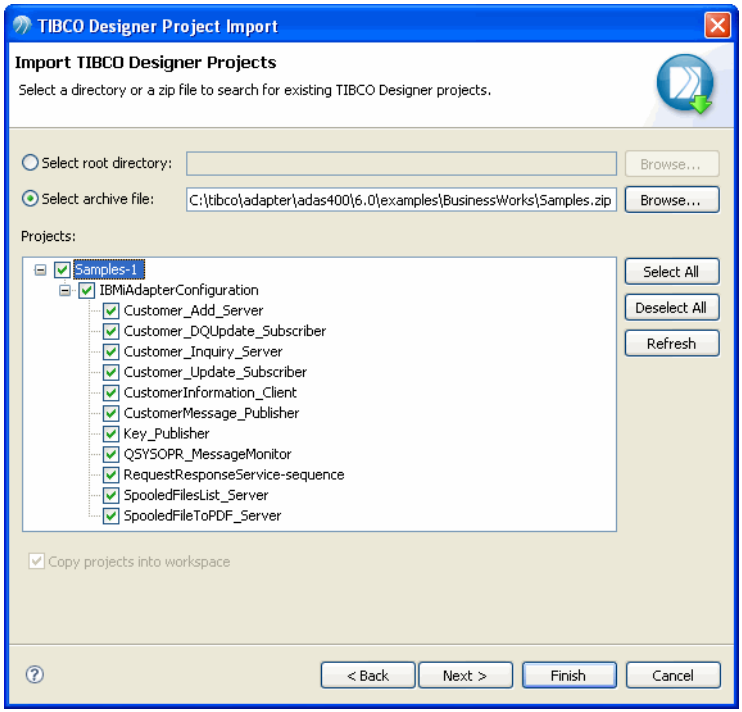
Importing TIBCO Designer Projects

The Adapter projects that are created using TIBCO Designer for the standalone component, TIBCO ActiveMatrix Adapter for IBM i, can also be imported into the TIBCO Business Studio environment.

To import a TIBCO Designer adapter project:

1. From the File menu, select **Import**.
2. In the Import dialog, select **TIBCO ActiveMatrix Adapter > Import TIBCO Designer Projects**. Click **Next**
3. Choose between **Select root directory** and **Select archive file**.
4. Click **Browse**. Navigate to the folder where the Adapter project or the archive file is located. Click **OK**.
5. In the Projects pane, select the checkbox for the project you want to import as well as the services contained in the project.

Figure 4 Import TIBCO Designer Project



The checkbox for the **Copy projects into workspace** is selected by default and the projects are copied to the current workspace.

6. Click **Finish**.

Differences in Imported Projects

When you import a TIBCO Designer adapter project, the following differences in the project and configuration should be noted:

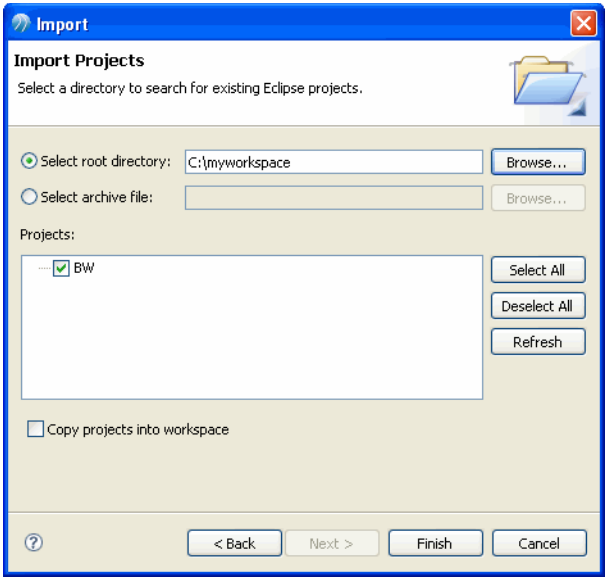
- The value of the Standard MicroAgentName field (located in the IBMiAdapterConfiguration->Monitoring tab in Designer) has been changed from `COM.TIBCO.ADAPTER.adas400.%%Deployment%%.%%InstanceId%%` to `COM.TIBCO.ADAPTER.adas400.%%Deployment%%.IBMiAdapterConfiguration`.
- The value of the Class MicroAgentName field (located in the IBMiAdapterConfiguration > Monitoring tab in Designer) has been changed from `COM.TIBCO.adas400.%%Deployment%%.%%InstanceId%%` to `COM.TIBCO.adas400.%%Deployment%%.IBMiAdapterConfiguration`.
- The following fields (features) are not imported:
  - Message Filter and version (in Configuration tab)

Importing Projects into Workspace

To import projects created in another workspace to the current workspace:

1. From the File menu, select **Import**.
2. In the Import dialog, select **General > Existing Projects into Workspace**. Click **Next**.
3. Choose between **Select root directory** and **Select archive file**.
   a. Click **Browse**.
   b. Navigate to the folder where the Adapter project or the archive file is located. Click **OK**.
4. In the Projects pane, select the projects you want to import.

Figure 5 Import Projects Created in Another Workspace



5. Select the **Copy projects into workspace** checkbox if you want to copy files belonging to the project into the current workspace.
6. Click **Finish**.

Chapter 3 Preparing IBM System i

This chapter explains how to prepare an IBM System i machine for use with the adapter.

Topics

- Loading a Library on an IBM System i Machine, page 12
- Adding the Library to a Library List, page 13
- Starting Host Server Daemons, page 14

Loading a Library on an IBM System i Machine

For the adapter to interact with an IBM System i machine, you need to load a saved file that contains pre-defined libraries to the IBM System i machine.

1. On the machine where the adapter is installed, start a command session. Change the directory of the command session to *ADAS400_HOME*`\examples`.
2. Start an FTP session in the command session to the IBM System i machine.
3. In the FTP session, change the transfer mode to binary using the command **`bin`**.
4. Transfer the saved file named `ADAS400.SAVF` from the *ADAS400_HOME*`\examples` directory to the IBM System i machine by entering the following command:

   `put ADAS400.SAVF` *library_name*`/ADAS400`

   where *library_name* is the name of the library where the saved file ADAS400 exists.
5. After the transfer is complete, sign on to the IBM System i machine and restore the example library `ADAS400` by entering the following command:

   `RSTLIB SAVLIB(ADAS400) DEV(*SAVF) SAVF(`*library_name*`/ADAS400)`

   where *library_name* is name of the library where the save file ADAS400 exists.

The example library is compatible with IBM i operating system version V5R4 or higher.

Adding the Library to a Library List

After restoring the ADAS400 library on the IBM System i machine, you need to add the library to library list every time you sign on.

Enter the following command from the IBM System i machine command menu:

```
ADDLIBLE ADAS400
```

Starting Host Server Daemons

To use the adapter's Request-Response Service and Request-Response Invocation Service, the remote command and distributed program call server daemon and the data queue server daemon must be started on the IBM System i machine.

Enter the following command from the IBM System i machine command menu:

```
STRHOSTSVR SERVER(*RMTCMD)
STRHOSTSVR SERVER(*DTAQ)
```

Chapter 4 Getting Started

This chapter describes the basic steps to configure and run the adapter service engine. Details for each step are provided later in the manual.

Topics

- Overview, page 16
- Prerequisites, page 17
- Configuring the Adapter Service Engine, page 18
- Configuring an ActiveMatrix SOA Project, page 22
- Running the Adapter Service Engine, page 23

Overview

The TIBCO ActiveMatrix Adapter Service Engine for IBM i (adapter service engine) enables the adapter to run in the ActiveMatrix environment as a container and to provide services.

The adapter service engine supports both the AE and ActiveMatrix endpoints. Adapter AE endpoints can be converted as ActiveMatrix endpoints that let the adapter service engine communicate seamless with other ActiveMatrix services. All adapter endpoint types are converted to Web Services Description Language (WSDL) and can be used for configuration in the ActiveMatrix environment.

The adapter service engine supports communication with TIBCO ActiveMatrix BusinessWorks and TIBCO ActiveMatrix BusinessWorks Service Engine using only the AE transport.

Prerequisites

Before starting the configuration, ensure that all required software have been installed and are operating correctly. For a list of required software, see *TIBCO ActiveMatrix Adapter Service Engine for IBM i Installation*.

You are supposed to know how to drag icons in TIBCO Business Studio and save projects. If you are not familiar with these topics, refer to the *Workbench User Guide*, which can be accessed from the Help > Help Contents menu option in TIBCO Business Studio.

See Preparing IBM System i on page 11 for instructions on how to prepare the IBM System i machine to work with the adapter service engine.

Configuring the Adapter Service Engine

TIBCO ActiveMatrix Adapter Service Engine for IBM i is configured with TIBCO Business Studio.

Configuration Task Sequence

To configure the adapter service engine, complete the following tasks:

- Task A, Start TIBCO Business Studio, page 18
- Task B, Create an Adapter Project, page 19
- Task C, Create an Adapter Instance, page 20
- Task D, Configure Connection for the Adapter Instance, page 20
- Task E, Import IBM i Business Objects, page 20
- Task F, Configure a Service, page 21

Task A Start TIBCO Business Studio

To start TIBCO Business Studio:

1. Execute one of the following platform-specific command:
   - On Microsoft Windows:

     select **Start > All Programs > TIBCO >** *AMX_HOME* **> TIBCO Business Studio 2.2 > TIBCO Business Studio**

     or

     run the executable from the command line

     AMX_HOME`\BusinessStudio\TIBCOBusinessStudio.exe`
   - On Unix:

     run *AMX_HOME*`/BusinessStudio/TIBCOBusinessStudio`
2. In the Workspace Launcher, accept the default workspace or browse to or create a new workspace. Click **OK**.
3. The first time you run TIBCO Business Studio, it displays a Welcome screen. Dismiss the screen by clicking the **Close** button ☒ in the title bar.

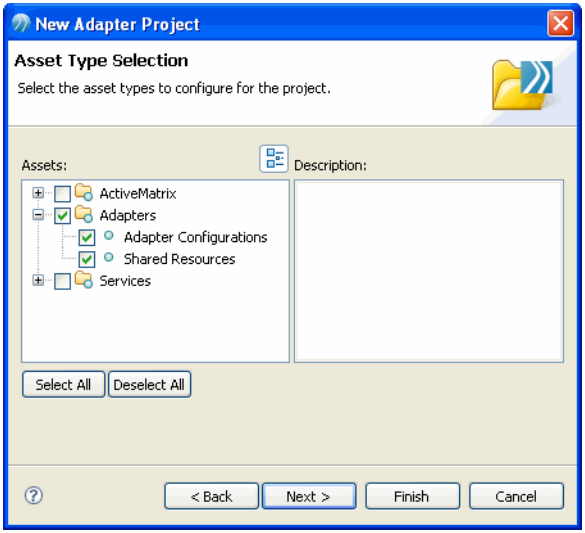
Task B Create an Adapter Project

A project contains the configuration files that define runtime options for the adapter.

To create an adapter project:

1. From the File menu, select **New** > **Project**.
2. In the New Project dialog, expand **TIBCO ActiveMatrix Adapters**, select **Adapter Project** as the wizard, and then click **Next**.
3. In the Project name field, enter a name for the project. Click **Next**.
4. In the Assets pane, select the asset types that are required for your project. This example uses the default setting, that is, only the Adapters asset type is selected.

Figure 6 Select Asset Types



5. Specify names for the selected assets.
   — To accept the default names, click **Finish**.
   — To use different names, click **Next**, and then enter new names in corresponding fields.

Task C Create an Adapter Instance

An adapter instance can contain Publisher, Subscriber, RPC Server, RPC Client, or all of them.

To create an IBM i adapter instance:

1. In the Project Explorer, right-click **Adapter Configurations**, and then select **New** > **Other** from the shortcut menu.
2. In the New dialog, expand **TIBCO ActiveMatrix Adapters** > **Configurations**, select **IBMi Adapter**, and then click **Next**.
3. In the IBM i Adapter Configuration dialog, click **Finish**. This creates an adapter instance named, by default, `IBMiAdapterConfiguration`. You can change the name in the Configuration section later.

Task D Configure Connection for the Adapter Instance

To configure the connection parameters:

1. Create and configure an IBM i Connection shared resource.

   See IBM i Connection Configuration Shared Resource on page 26 for details.
2. Deploy the IBM i connection.
   a. Go to the adapter instance editor.
   b. Click the **Adapter Options** tab.
   c. In the Design-time Connection section, click **Connection Configuration**.
   d. In the Connection Configuration dialog, select the IBM i connection that you want to use. Click **OK**.
   e. Select the **Use Design-Time Connection Configuration for Run-Time** checkbox.
3. Specify the work library and library list for the adapter.
4. Save the configuration.

Task E Import IBM i Business Objects

1. From the File menu, select **Import**.
2. In the Import dialog, expand **TIBCO ActiveMatrix Adapter** > **Metadata**, select **IBM i Business Object**, and then click **Next**.
3. Configure the importing information in the Import Wizard. See Importing IBM i Business Object on page 28 for details.

Task F Configure a Service

The next step is to configure an adapter service that performs the publication, subscription, or request-response data passing operations for the adapter service engine.

1. In the adapter instance editor, click the **Adapter Services** tab.
2. Click the corresponding buttons to add the services you want to use. See Adapter Services Tab on page 39 for how to configure different services.
3. From the File menu, select **Save** to save the project information.

Configuring an ActiveMatrix SOA Project

After configuring the adapter project, you need to create an SOA project and generate service assembly file and deployable archive.

See Configuring ActiveMatrix SOA Project on page 53 for instructions on how to do that.

Running the Adapter Service Engine

To run the adapter service engine, you need to deploy the service assembly in TIBCO ActiveMatrix Administrator. See Deploying the Service Assembly on page 61 for instructions on how to do that.

Chapter 5 Configuring Adapter Instance

This chapter describes the options available for configuring an adapter instance and for adding services. All configuration tasks are done using TIBCO Business Studio.

Topics

- IBM i Connection Configuration Shared Resource, page 26
- IBM i Business Object, page 28
- IBM i Adapter Instance Configuration, page 35

IBM i Connection Configuration Shared Resource

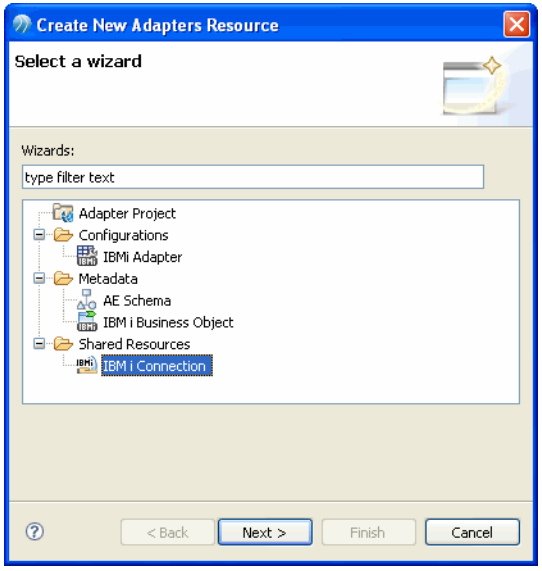
For a newly created adapter project, you need to create an IBM i connection and configure it to access the IBM System i machine.

Creating an IBM i Connection

To create an IBM i connection:

1. Select a project in the Project Explorer, and then from the File menu select **New > Adapter Resources**.
2. In the Create New Adapters Resource dialog, expand **Shared Resources**, select **IBM i Connection**. Click **Next**.

Figure 7 Create an IBM i Connection



3. Select the parent folder for the IBM i Connection shared resource. For example, the `Shared Resources` folder under the adapter project folder. Click **Finish**.

Configuring an IBM i Connection

An IBM i Connection is configured in the IBM i Connection Shared Resource Editor.

1. Double-click the connection configuration file under the adapter project folder in the Project Explorer.
2. In the IBM i Connection Shared Resource Editor, specify the connection parameters of the connection.

   You can also click the **Set Bindings** button on the right-hand of the title bar to bring up the Binding Editor and set values from substitution variables.

Figure 8 IBM i Connection Shared Resource Editor

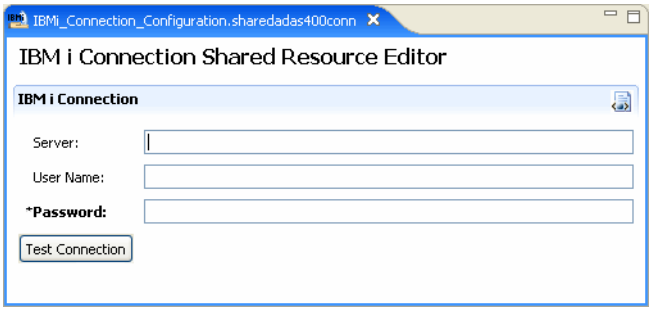


Table 3 IBM i Connection Configuration Parameters

Field	Description
Server	The IP address or host name of the IBM System i machine for this connection.
User Name	The IBM System i user profile name. A user profile is used to connect to the IBM System i machine when configuring the adapter at design time. The user profile should have been authorized to access objects that will be used in service configurations. Also, the user profile should have full authority for the library specified in the Work Library field.
Password	the corresponding password of the user name that will be used to access the IBM System i machine.

3. Click the **Test Connection** button to verify the configuration.
4. Save the configuration.

IBM i Business Object

TIBCO ActiveMatrix Adapter Service Engine for IBM i introduces the concept of a *Business Object (BO)*. In the simplest sense, a BO is the representation of the data model for the business entities such as invoice, orders, customers, or vendors.

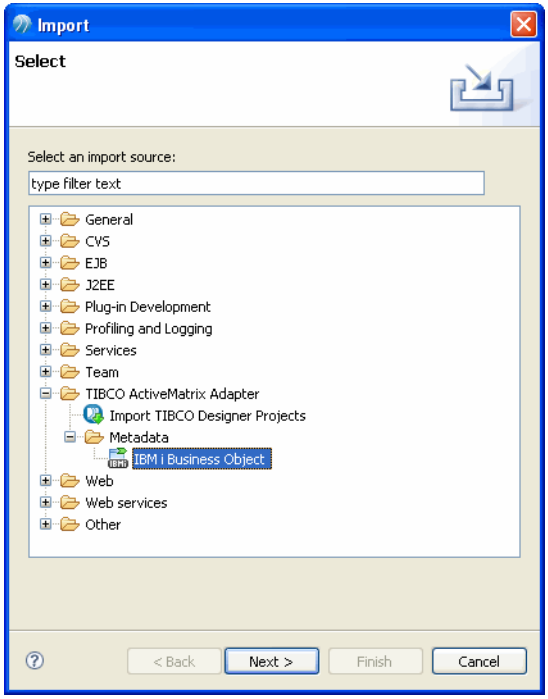
The BO also encapsulates the data, its attributes, types, and functions as well as the operations and relationships between other BOs. The schema that is downloaded can be used by adapter services in an adapter project.

Importing IBM i Business Object

To import IBM i business object:

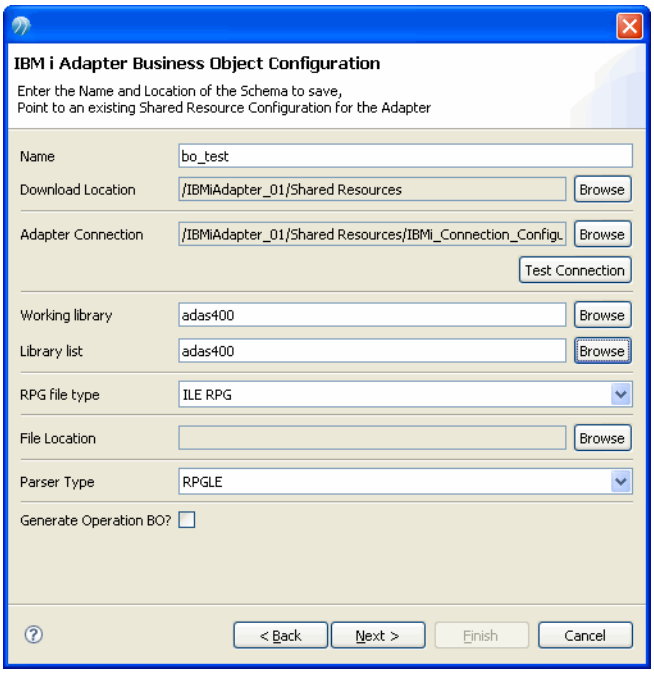
1. From the File menu, select **Import**.
2. In the Import dialog, expand **TIBCO ActiveMatrix Adapter** > **Metadata**, select **IBM i Business Object**, and then click **Next**.

Figure 9 Import IBM i Business Object



3. Configure the general information in the Import Wizard.
   a. In the name field, enter the name of the file to save the schema.
   b. In the Download Location filed, browse to select the location to save the schema. For example, *proj*/`Shared Resources`.
   c. In the Adapter Connection field, browse to select a pre defined IBM i Connection shared resource. Click the **Test Connection** button to verify the connection configuration.
   d. Specify the Working library and Library list.

Figure 10 IBM i Business Object Import Wizard



4. Specify the relevant fields in the Import Wizard based on the way to import Business Objects.
   — To import BO from a source file, select the RPG file type.
   — To import BO from an XML schema file, click the **Browse** button next to the File Location field to choose the `.xsd` file of the XML schema.
   — To import BO from object parameters, select a parser from the Parser Type drop-down list. The parser type must match the source file type on the IBM

System i machine. Otherwise, the business objects cannot be parsed correctly.

— To import BO from objects, you need not do specific configuration in the Import Wizard.

5. If the BO is to be imported from parameters, configure the **Generate Operation BO** option. This option only affects the Subscriber and RPC Server.

   — For Subscriber, clear the **Generate Operation BO** checkbox. In this way, the generated BO will be an ordinary BO.

   — For RPC Server, select the **Generate Operation BO** checkbox. In this way, the generated BO will be an operation BO.

Ordinary BOs are used for Publisher and Subscriber. Operation BOs are used for RPC Server and RPC Client.

You can add an operation to an Operation BO in the IBM i Adapter Business Object Editor, while you cannot do that to an ordinary BO. See Operation BO on page 32 for details.

6. Click **Next**.

7. Select the business objects that you want to retrieve from the IBM System i.

   — To import BO from objects, search in the **Object Group** under the Applications tree.

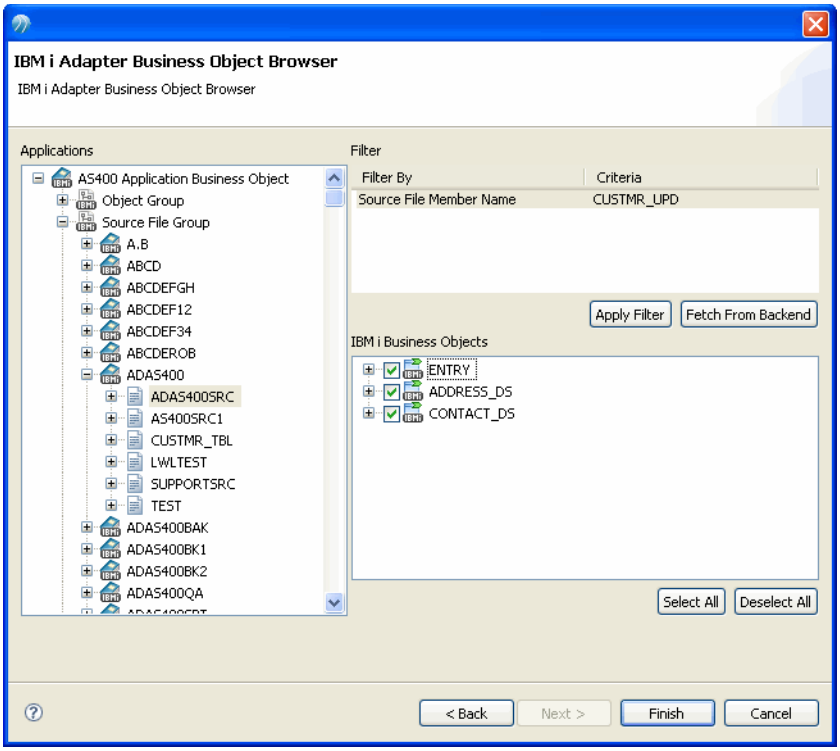
   — To import BO from a source file, search in the **Source File Group** under the Applications tree.

   — To import BO from an XML schema file, search in the **XML Schema File Group** under the Applications tree.

   — To import BO from parameters, search in the **Object Parameter Group** under the Applications tree.

   When importing from a source file or from parameters, you also need to enter the member file name as the criteria in the Filter table and then click the **Fetch From Backend** button, as shown in Figure 11.

Figure 11 Select from the Application Tree



8. Click **Finish**.

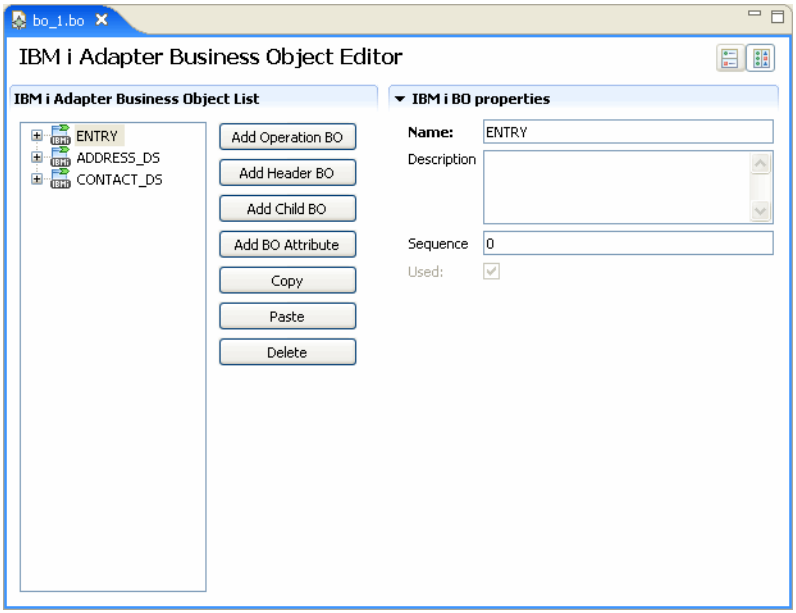
After successfully importing BOs, a business object file (`.bo`) will be automatically generated in the adapter project.

Editing IBM i Business Objects

The business objects can be browsed or edited in the IBM i Adapter Business Object Editor.

Double-click the business object file under the adapter project folder in the Project Explorer to launch the IBM i Adapter Business Object Editor.

Figure 12 IBM i Adapter Business Object Editor



You can use the buttons next to the IBM i Adapter Business Object List to perform the following tasks:

- adding an operation BO
- adding a header BO
- adding a child BO
- adding a BO attribute
- deleting a selected object

Further configurations may be required depending on the type of the selected business object.

Operation BO

Operation BOs are used for PRC Server and RPC client. A DQWrite operation, ProgramCall operation, or OutBound operation can be added to an Operation BO.

- A DQWrite Operation is used for RPC Server with Data Queue Write service type.
- A ProgramCall Operation is used for RPC Server with Program Call service type.

If the program source is parsed successfully, the parameters are displayed as a schema in the editor. Otherwise, the program parameters can be defined manually.

- An OutBound Operation is used for RPC Client with Outbound Message service type.

Converting Business Objects into AESchemas

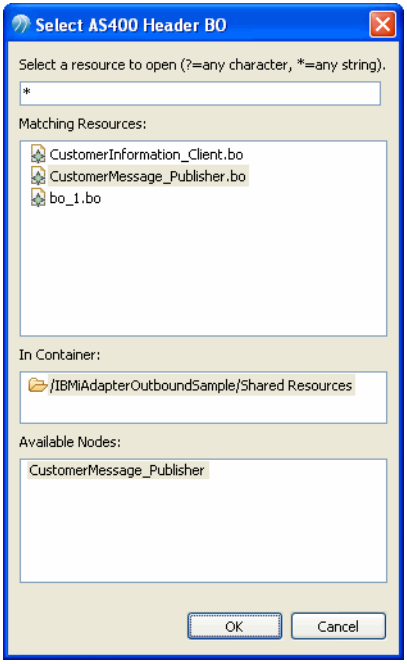
After generating a business object file (`.bo`), you need to convert the BOs into AESchemas. The conversion is performed during the configuration of the adapter instance.

For most types of adapter services, there is an option named Class Reference From BO in the Schema section. To do the conversion:

1. Select **Browse** from the Class Reference From BO drop-down list.
2. Select a `.bo` file from the Matching Resources list and select an adapter project from the In Container list. Then select a schema from the Available Nodes list.

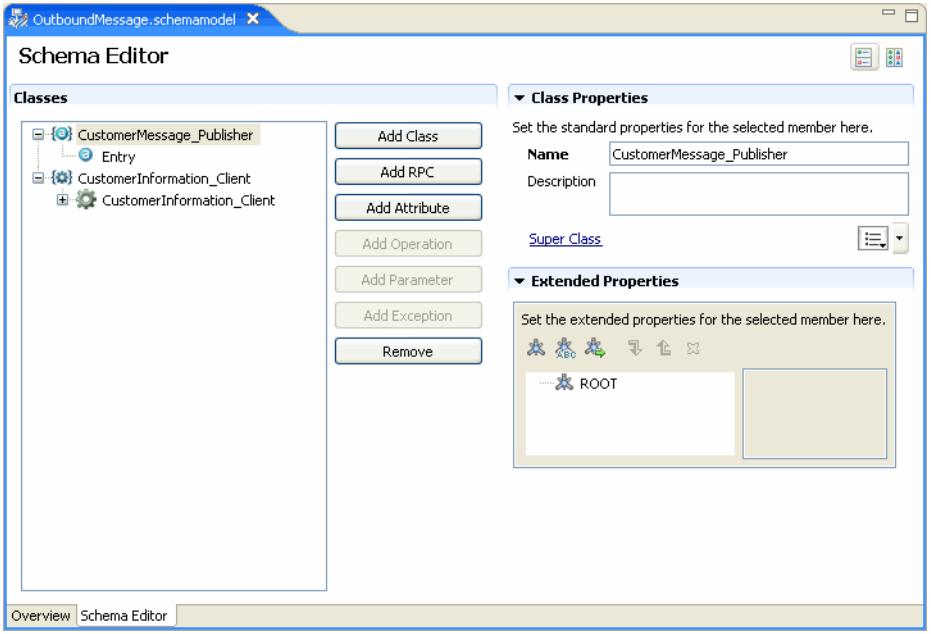
   Ordinary BOs are only available for Publisher and Subscriber. Operation BOs are only available for RPC Server and RPC Client.

Figure 13 Select BO



3. Click **OK**. The converted schema class file is displayed in the Class Reference field.
4. Click the **Class Reference** field to launch the Schema Editor, in which you can browse or edit the schema class.

Figure 14 Schema Editor



IBM i Adapter Instance Configuration

This section describes the options available for configuring an adapter instance and for adding services.

- Configuration Tab, page 35
- Adapter Options Tab, page 37
- Adapter Services Tab, page 39
- Logging Sinks Tab, page 49

Configuration Tab

In the Configuration tab, you can provide basic configuration information in several sections for the adapter instance.

Table 4 Field Description of Configuration Tab

Field	Description
IBM i Adapter Configuration	
Adapter Name	Name of the adapter. The value of this field cannot be changed.
Instance Name	Use the default name or replace it with a name of your choice. See Guidelines for Choosing an Instance Name on page 37.
Description	A short description of the adapter configuration.
Startup	
Show Startup Banner	The startup banner displays the runtime adapter version, the infrastructure version on which the adapter service engine is built, and the copyright information in the console window when the adapter service engine is started. The content of the start banner is pre defined and cannot be changed. Clear this checkbox if you do not want to display the startup banner.
Multithreading Pool	
Inbound Thread Count	Number of threads to process requests inbound to the adapter. Request-Response Service and Subscription Service process inbound requests to the adapter. The thread count can be used to control the number of inbound requests to be processed by the adapter concurrently.

Table 4 Field Description of Configuration Tab (Cont'd)

Field	Description
Monitoring	
Enable Standard MicroAgent	This checkbox allows you to turn on or turn off the standard TIBCO Hawk Microagent.
Standard MicroAgent Name	The name for the standard microagent that will be registered with the TIBCO Hawk system. Click the **Set Bindings** button on the title bar to launch the Binding Editor. In the Binding Editor, you can pick substitution variables to bind with properties Standard MicroAgent Name. In most cases, the default value is used. The InstanceId variable need not be set because it is automatically set at runtime by the runtime adapter.
Standard MicroAgent Timeout(ms)	The timeout value for the standard microagent in milliseconds.
Enable Class MicroAgent	This checkbox allows you to turn on or turn off the instance or class-specific standard TIBCO Hawk Microagent. Click the **Set Bindings** button on the title bar to launch the Binding Editor. In the Binding Editor, you can pick substitution variables to bind with properties Enable Class MicroAgent.
Class MicroAgent Name	The name for the class microagent that will be registered with the TIBCO Hawk system. In most cases, the default value is used.
Class MicroAgent Timeout(ms)	The timeout value for the class microagent in milliseconds.
Default MicroAgent Session	The name of the TIBCO Rendezvous session that will be used by the standard, class, and custom microagents. Do not change the session name and the corresponding session, which are automatically generated by TIBCO Business Studio. However, you can double-click the `HawkSession.rvtransport` under the *instancename*_`genresources` folder in the Project Explorer to modify the session parameters if required. Ensure that you have set the correct parameter value for the substitution variables that correspond to the TIBCO Hawk configuration. Otherwise, the microagents will not appear in the TIBCO Hawk Display.

Guidelines for Choosing an Instance Name

- The entire instance name must be less than 80 characters.
- The following characters cannot be used in an instance name: `/`, `\`, `<`, `>`, `|`, `*`, `%`, `:`, `+`, `#`, `?`, and `"`.
- The space character and dot character cannot be used to begin or end an instance name.
- An instance name cannot use substitution variables.
- An instance name must be unique with respect to other adapter instances for the same adapter in the project.

  The same instance name can be used to name an adapter instance for a different adapter in the same project. For example, a Lotus Notes adapter instance named TEST and a Siebel adapter instance named TEST can coexist in the same project.

  Each instance name must be unique per adapter within a project even if each instance is defined in a different folder. That is, configuring same-named adapter instances in different folders will not make their names unique.

When you create an adapter instance, the palette automatically creates several resources for it. The names of these resources derive from the name of the instance to which they belong. Changing the adapter instance name results in an automatic regeneration of the resources names. If you manually modify any resource name, that particular name will *not* be automatically regenerated the next time you rename the adapter instance.

Adapter Options Tab

Configure the connection information in this tab.

Table 5 Fields of Adapter Options Tab

Field	Description
Design-time Connection	
Connection Configuration	A connection configuration for the design-time. An IBM i Connection shared resource must exist before you can refer to it. See Configure Connection for the Adapter Instance on page 20 for details.
Use Design-Time Connection Configuration for Run-Time	Select this checkbox to apply this connection at runtime. If not selected, you need to specify runtime connection parameters in the Run-time Connection section.

Table 5 Fields of Adapter Options Tab (Cont'd)

Field	Description
Generate Utility Programs	Select this checkbox to generate utility programs (PUBLISHER, RPCCLIENT, RPCSERVER, and SUBSCRIBER) on the IBM System i machine. If not selected, the utility programs won't be generated.
Working Library and Library List	
Work Library	Name of the library that will be used to generate temporary object lists and as a default library when generating any adapter services configuration related objects.
Library List	A library list on the IBM System i machine is equivalent to `PATH` on Microsoft Windows platform or `LD_LIBRARY_PATH` on UNIX platforms. Clicking the Browse resources button next to this field will bring up a popup window with list of libraries. You can select multiple libraries from the list by pressing the Ctrl key. Select libraries in the popup window in the order that the libraries appear in the library list.
Run-time Connection	
Initial Connections	The number of connections that should be made to the adapter when the adapter starts up.
Maximum Connections	The maximum number of connections that can be made to the adapter.
Connection Idle Time (milliseconds)	The idle time allowed for a connection.
Reset Count	This option is not applicable for the TIBCO ActiveMatrix Adapter Service Engine for IBM i.
Retry Total	The total number of reconnection attempts to make before the runtime adapter or adapter service is stopped. A value of -1 means reconnection attempts will continue indefinitely.
Reconnection Attempts	This option is not applicable for the TIBCO ActiveMatrix Adapter Service Engine for IBM i.
Retry Interval	This option is not applicable for the TIBCO ActiveMatrix Adapter Service Engine for IBM i.
Connection Configuration	A connection configuration for the runtime. An IBM i Connection shared resource must exist before you can refer to it. See Configure Connection for the Adapter Instance on page 20 for details.

Table 5 Fields of Adapter Options Tab (Cont'd)

Field	Description
Library List	Clicking the Browse resources button next to this field will bring up a popup window with the list of libraries. You can select multiple libraries from the list by pressing the Ctrl key. You should select libraries in this popup window in the order that the libraries should appear in the library list.
Initial Program To Run	Initial program is a command that will be called each time a new connection is established with the IBM System i machine. For example, you can enter **`ADDLIBLE ADAS400`** in this field to add the `ADAS400` library to library list upon the establishment of a new connection. This option is also useful if you want to run additional commands for new recreated connections after the reconnection is done. The initial program should not contain any runtime parameters.

Adapter Services Tab

This tab provides options for adding and configuring adapter services:

- Publisher, page 39
- Subscriber, page 41
- RPC Server, page 43
- RPC Client, page 47

Publisher

Publisher sends messages from IBM System i machine to applications configured for the TIBCO environment. Click **Add Publisher** and then specify values for the fields listed in Table 6.

Table 6 Publisher Options

Field	Description
Configuration	
Name	Name of the service. You can use the default name or replace it with a name of your choice. A service name must use alphanumeric characters. An underscore (_) character can be used. The entire instance name must be less than 80 characters. The space character cannot be used in a service name. A service name cannot use substitution variables.

Table 6 Publisher Options (Cont'd)

Field	Description
Description	Additional information about the Publication Service being configured. This field is optional.
Schema	
Class Reference	The AE Schema that is automatically generated from the selected BO.
Class Reference From BO	Convert an IBM i Business Object into AESchema. Browse to select business object child from the previously imported BOs. See Converting Business Objects into AESchemas on page 33 for details.
Queue Configuration	
Data Queue for Publish	The data queue name (*DTAQ object) that will be used for publishing the message from an application running on the IBM System i machine. You can assign a substitution variable to this field. The format of the data queue name is *library_name/data_queue_name*. Clicking the Browse resources button next to this field brings up a popup window with the list of libraries. You can expand each library to view list of available data queue (`*DTAQ`) objects. You can also select a library and create a new data queue in it.
Is Data Queue Shared?	Whether a data queue specified for a service is shared among multiple services. See Guidelines for Using Shared Data Queue on page 41 for details.
Is Peek/Read Used?	If selected, you can customize the polling interval of the Publication Service to poll an IBM System i machine for messages in the data queue through the property named `adas400.pollingInterval` in the properties file. If not selected, the Publication Service will be blocked when there are no messages in the data queue.
Is Publisher Key Used?	If selected, Publication Service uses a specified key value to query a keyed data queue, only those messages that have a matching key will be published. If not selected, messages are published regardless of the key.
Publisher Key	Value of the key used to query the keyed data queue.
Generate Source Type	When message schema is defined for the publisher, a copy source member containing message schema will be generated in a library on the IBM System i machine. This field specifies the type of generated source member. The generated source type can only be ILE `RPG` in the current release.

Table 6 Publisher Options (Cont'd)

Field	Description
Generate in Source File	The source file in which the source member for message schema will be generated. Clicking the Browse resources button next to this field brings up a popup window with list of libraries. You can expand each library to view the list of available file (`*FILE`) objects. You can also select a library and create a new source file in it.
Generate Member of Name	Name of the source member that will be generated.
UPLOAD	Click this button to upload the message schema to the IBM System i machine and generate the RPGLE source for the schema.

Guidelines for Using Shared Data Queue

- If a data queue is marked as shared, the data queue can be shared with other Publication Service and Request-Response Invocation Service.

  Each message entry in a shared data queue must contain a routing header. If a message is published to the shared data queue using PUBLISHER program, which is generated by the adapter at design-time, the routing header information is automatically added to the message.
- A data queue marked as non-shared cannot be shared with other services.
- The application program running on IBM System i machine publishes messages to a non-shared data queue through system APIs.

Subscriber

Subscriber sends messages from the applications configured for the TIBCO environment to the IBM System i machine. Click **Add Sub Service** and then specify values for the fields listed in Table 7.

Table 7 Subscriber Options

Field	Description
Configuration	
Name	Name of the service. You can use the default name or replace it with a name of your choice. A service name must use alphanumeric characters. An underscore (_) character can be used. The entire instance name must be less than 80 characters. The space character cannot be used in an instance name. A service name cannot use substitution variables.
Description	Additional information about the adapter service being configured. This field is optional.

Table 7 Subscriber Options (Cont'd)

Field	Description
Schema	
Class Reference	The AE Schema that is automatically generated from the selected BO.
Class reference from BO	Convert an IBM i Business Object into AESchema. Browse to select business object child from the previously imported BOs. See Converting Business Objects into AESchemas on page 33 for details.
Services Category Configuration	
Service Type	There are two types of Subscription Service, as follows: **Data Queue Write** Writes messages to data queues on IBM System i machine either asynchronously or synchronously. **Program Call** Invokes application programs running on the IBM System i machine.
Sub Queue Configuration (This section is available only for the Data Queue Write service type.)	
Data Queue Name	Name of the data queue (*DTAQ object) that sends the request message to the IBM System i machine. You can assign a substitution variable to this field. The format of the data queue name is *library_name/data_queue_name*. Clicking the Browse resources button next to this field will bring up a popup window with a list of libraries. You can expand each library to view the list of available data queue (*DTAQ) objects. You can also select a library and create a new data queue in it.
Generate Source Type	When message schema is defined for the service, a copy source member containing message schema is generated in a library on the IBM System i machine. This field accepts the type of generated source member. The generated source type can only be ILE RPG in the current release.
Generate in Source File	The name of the source file where message schema will be generated. The format of source file name is *library_name/source_file_name*. Clicking the Browse resources button next to this field will bring up a popup window with list of libraries. You can expand each library to view list of file (`*FILE`) objects available in that library.
Generate Member of Name	Name of the source member to be generated. The source member will contain message schema.
UPLOAD	Click this button to upload the message schema to the IBM System i machine and generate the ILE RPG source for the schema.

Table 7 Subscriber Options (Cont'd)

Field	Description
Sub Program Configuration (This section is available only for the Program Call service type.)	
Program Object	Name of program object (`*PGM` object). The format of program object name is *library_name/program_name*. Clicking the Browse button next to this field will bring up a popup window with list of libraries. You can expand each library to view the list of available program (`*PGM`) objects.
Source File	Name of the source file that contains the program source member. The format of the source file name is *library_name/source_file_name*. Clicking the Browse button next to this field will bring up a popup window with list of libraries. You can expand each library to view list of available file (`*FILE`) objects. You can also select a library and create a new source file in it.
Source Member	Name of program source member.

RPC Server

Request-Response Service is initiated by an application running in the TIBCO environment. Click **Add RPC Server** and then specify values for the fields listed in Table 8.

Table 8 RPC Server Options

Field	Description
Configuration	
Name	Name of the RPC Service. A service name must use alphanumeric characters. An underscore (_) character can be used. The entire instance name must be less than 80 characters. The space character cannot be used in an instance name. A service name cannot use substitution variables.
Description	Additional information about the adapter service being configured. This field is optional.
Schema	
Class Reference	The AE Schema that is automatically generated from the selected BO.
Class Reference From BO	Convert an IBM i Business Object into AESchema. Browse to select business object child from the previously imported BOs. See Converting Business Objects into AESchemas on page 33 for details.

Table 8 RPC Server Options (Cont'd)

Field	Description
Service Category Configuration	
Service Type	There are four types of Request-Response Service, as follows: **Data Queue Write** Writes messages to data queues on IBM System i machine either asynchronously or synchronously. **Program Call** Invokes application programs running on the IBM System i machine. **Spooled File List** Retrieves list of spooled files from the IBM System i machine. **Spooled File to PDF** Retrieves a spooled file from the IBM System i machine and convert it to PDF format. The current release supports conversion of only *SCS type spooled files to PDF format.

Table 8 RPC Server Options (Cont'd)

Field	Description
Server Queue Configuration (This section is available only for the Data Queue Write service type.)	
Request Data Queue	Name of the data queue (*DTAQ object) that sends the request message to the IBM System i machine. A substitution variable can be assigned to this field. The format of the data queue name is *library_name/data_queue_name*. The request queue can be either a keyed data queue or a none-keyed data queue. You can define your own key and put it in the request message. Clicking the Browse button next to this field to bring up a popup window with a list of libraries. You can expand each library to view list of available data queue (*DTAQ) objects. You can also select a library and create a new data queue in it.
Reply Data Queue (optional)	Name of the data queue (*DTAQ object) that reads the reply message from the IBM System i machine. A substitution variable can be assigned to this field. The format of the data queue name is *library_name/data_queue_name*. The reply data queue must be a keyed data queue. The reply data queue should not be the same as the request data queue. Clicking the Browse resources button next to this field will bring up a popup window with a list of libraries. You can expand each library to view list of available data queue (*DTAQ) objects.
Timeout (milliseconds) (optional)	The timeout value in milliseconds for Request-Response Service. By default, the timeout value is 30000 milliseconds (30 seconds).
Are Data Queues Shared?	Whether data queues specified for a Request-Response Service are shared among multiple services. If selected, the adapter will generate a unique 112 character key for request-response operation at runtime. If not selected, the key for request-response operation is user specified.
Generate Source Type	When message schema is defined for the Request-Response Service, a copy source member containing message schema is generated in a library on the IBM System i machine. This field specifies the type of generated source member. The generated source type can only be ILE RPG in the current release.
Generate in Source File	The source file where message schema will be generated. The format of the source file name is *library_name/source_file_name*. Clicking the Browse resources button next to this field will bring up a popup window with list of libraries. You can expand each library to view list of available file (`*FILE`) objects.

Table 8 RPC Server Options (Cont'd)

Field	Description
Generate Member of Name	Name of the source member that will be generated. The source member will contain message schema.
UPLOAD	Click this button to upload the message schema to the IBM System i machine and generate the ILE RPG source for the schema.
Server ProgramCall Configuration (This section is available only for the Program Call service type.)	
Program Object	Name of the program object (`*PGM` object). The format of program object name is *library_name/program_name*. Clicking the Browse button next to this field to bring up a popup window with list of libraries. You can expand each library to view list of program (`*PGM`) objects available in that library.
Source File	Name of the source file that contains the program source member. The format of source file name is *library_name/source_file_name*. Clicking the Browse button next to this field to bring up a popup window with list of libraries. You can expand each library to view list of available file (`*FILE`) objects.
Source Member	Name of program source member.
Server Spooled File To PDF Configuration (This section is available only for the Spooled File To PDF service type.)	
Paper Size	The PDF paper size.
Paper Orientation	The PDF paper orientation.
Unit for Margins	The unit of measure for page margins of the PDF.
Top Margin /Bottom Margin /Left Margin /Right Margin	The page margins of the PDF.

RPC Client

RPC Client responds to the messages from applications configured for the TIBCO environment. Click **Add RPC Client** and then specify values for the fields listed in Table 9.

Table 9 RPC Client Options

Field	Description
Configuration	
Name	Name of the service. You can use the default name or replace it with a name of your choice. A service name must use alphanumeric characters. An underscore (_) character can be used. The entire instance name must be less than 80 characters. The space character cannot be used in an instance name. A service name cannot use substitution variables.
Description	Additional information about the adapter service being configured. This field is optional.
Schema	
Class Reference	The AE Schema that is automatically generated from the selected BO.
Class Reference From BO	Convert an IBM i Business Object into AESchema. Browse to select business object child from the previously imported BOs. See Converting Business Objects into AESchemas on page 33 for details.
Service Category Configuration	
Service Type	Type of the RPCC Service. **Monitor Message** The service is used as a message monitoring service. If you plan to call an application program which is likely to generate messages which require a reply (error or otherwise) on a message queue, you can configure a Monitor Message type of Request-Response Invocation Service and associate this service with program call service. **Outbound Message** The service is used to process the outbound request-response operation initiated by applications running on an IBM System i machine.
Client Monitor Message Configuration (This section is available only for the Monitor Message service type.)	
Message Queue	The name of a message queue (`*MSGQ` object). Format of the message queue name is *library_name/message_queue_name*. Clicking the Browse button next to this field to bring up a popup window with a list of libraries. You can expand each library to view the list of available message queue (`*MSGQ`) objects.

Table 9 RPC Client Options (Cont'd)

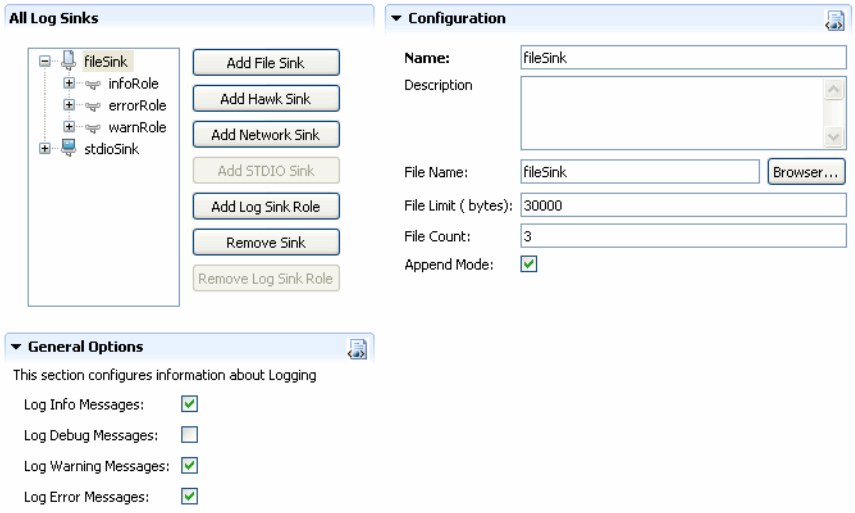
Field	Description
Timeout (milliseconds)	The timeout value in milliseconds for RPC Client. By default, the value for timeout is 30000 milliseconds (30 seconds).
Client Queue Configuration (This section is available only for the Outbound Message service type.)	
Request Queue	Name of the data queue (`*DTAQ` object) that sends the request message from an application running on the IBM System i machine. The request data queue must be a keyed data queue. The format of the data queue name is *library_name/data_queue_name*. Clicking the Browse button next to this field to bring up a popup window with a list of libraries. You can expand each library to view the list of available data queue (`*DTAQ`) objects. You can also select a library and create a new data queue in it.
Reply Queue	Name of the data queue (`*DTAQ` object) that sends the reply message to an application running on the IBM System i machine. The reply data queue must be a keyed data queue and be different from the request data queue. The format of the data queue is *library_name/data_queue_name*. Clicking the Browse button next to this field to bring up a popup window with a list of libraries. You can expand each library to view the list of available data queue (`*DTAQ`) objects. You can also select a library and create a new data queue in it.
Timeout (milliseconds)	The timeout value in milliseconds for Request-Response Invocation Service. By default, the value for timeout is 30000 milliseconds (30 seconds).
Are Data Queues Shared?	Whether the data queues specified for the service are shared among multiple services. If this checkbox is selected, data queues can be shared with other Publication Service and Request-Response Invocation Service. The length of the key for data queues should be 112 characters. For shared data queues, each message entry in the request data queue must contain a routing header. A message should be published to the request data queue using RPCCLIENT program generated by the adapter at design-time. If a message is published to the shared data queue using RPCCLIENT program, the routing header information is automatically added to the message. If this checkbox is not selected, the data queues cannot be shared with other services. The length of key for data queues can be any key length and the message entry can be in any format. A message should not be published to a non-shared data queue using RPCCLIENT program. The application program running on IBM System i machine must send messages to a non-shared data queue using system APIs.

Table 9 RPC Client Options (Cont'd)

Field	Description
Generate Source Type	The type of generated source member. When message schema is defined for the publisher, a copy source member containing message schema is generated in a library on the IBM System i machine. The generated source type can only be ILE RPG in the current release of the adapter.
Generate in Source File	Name of the source file in which source member for message schema is generated. The format of the source file name is *library_name/source_file_name*. Clicking the Browse resources button next to this field will bring up a popup window with a list of libraries. You can expand each library to view the list of available file (*FILE) objects.
Generate Member of Name	Name of the source member that will be generated.
UPLOAD	Click this button to upload the message schema to the IBM System i machine and generate the ILE RPG source for the schema.

Logging Sinks Tab

Figure 15 Log Sinks



Specify the output destination (sinks) for the trace messages in the Log Sinks tab.

All Log Sinks

In this section, you can perform following tasks:

- Add a log sink
- Add a role to a log sink
- Remove a log sink
- Remove a role from a log sink

By default, there are a File sink and a STDIO sink. You can add more sinks. The available sink types are: File, Hawk, Network, and STDIO.

General Options

Trace messages of the selected levels will be collected in the log sinks.

There are three types of logs (log sinks) that you can configure to hold trace messages, corresponding to three levels (roles) of trace messages, Information, Warning, and Error. A fourth level of trace messages, Debug, is reserved and should not be enabled unless requested by the TIBCO Product Support Group. To hold Debug trace messages will write a lot of information to the log file and significantly degrades the performance of the adapter service engine.

Turning on the levels can affect the performance of the adapter service engine. Therefore, it is recommended to turn on the required levels only.

Configuration

The configuration options varies with the sink types.

Table 10 Field Description of Log Sinks Configuration

Field	Description
File Sink	
Name	The name of the log sink.
Description	Description about the log sink.
File Name	The path and name of the trace file.
File Limit (bytes)	Maximum size of the file in bytes. The default value is 30000 bytes.
File Count	The number of rollover files. The default value is 3.

Table 10 Field Description of Log Sinks Configuration (Cont'd)

Field	Description
Append Mode	If checked, traces are appended to the existing file at startup. If unchecked, the existing file is overwritten at startup if one of the same name exists.
Hawk Sink	
Name	The name of the log sink.
Description	Description about the log sink.
MicroAgent Name	Name of the microagent for traces from this Hawk sink. The Hawk sink uses the hawk session, created and used by the adapter for monitoring purposes, to send tracing messages to the TIBCO Hawk monitor or Display.
Network Sink	
Name	The name of the log sink.
Description	Description about the log sink.
Subject	The subject on which the trace messages need to be published. The Network sink is used to publish tracing message on TIBCO Rendezvous.
Session Reference	Browse to select one of the previously defined sessions.
STDIO Sink	
Name	The name of the log sink.
Description	Description about the log sink.
Output Stream	The option to write to `out` or `error`.

Chapter 6 Configuring ActiveMatrix SOA Project

This chapter describes the process of creating an adapter component, adding services and references to the adapter component, and creating a service assembly.

Topics

- Creating an ActiveMatrix SOA Project, page 54
- Generating WSDL Files, page 55
- Configuring Composites, page 56
- Creating Service Assembly and Service Assembly Archive, page 60

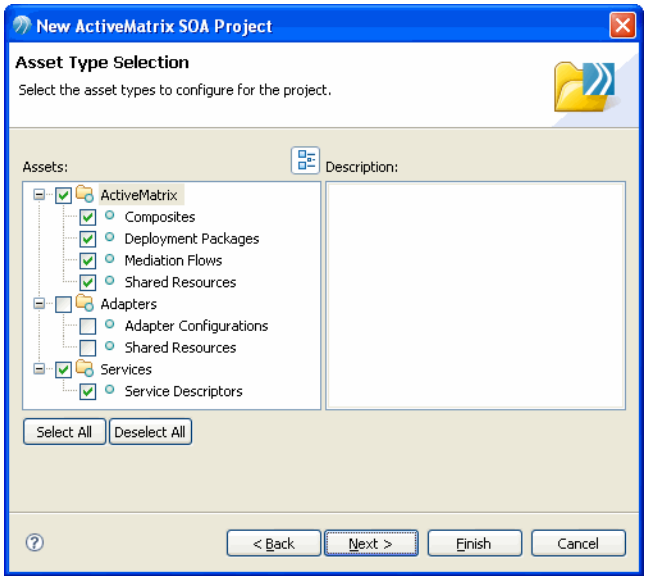
Creating an ActiveMatrix SOA Project

ActiveMatrix SOA projects contain the assets that you can develop with TIBCO Business Studio. Before creating any type of asset you must first create an ActiveMatrix SOA project.

To create an ActiveMatrix SOA project:

1. From the File menu, select **New** > **Project**.
2. In the New Project dialog, expand **TIBCO ActiveMatrix**, select **ActiveMatrix SOA Project**. Click **Next**.
3. In the Project name field, enter a name for the project, for example, `soa_proj01`.
4. Select **Basic SOA Project** from the Available Templates list. The Basic SOA Project template creates a project with an empty composite. Click **Next**.
5. In the New ActiveMatrix SOA Project dialog, select the asset types that are required for your project.
6. Specify names for selected assets.
   - To accept the default names, click **Finish**.
   - To use different names, click **Next**, and then enter new names in corresponding fields.

Figure 16 Create an ActiveMatrix SOA Project



Generating WSDL Files

TIBCO ActiveMatrix services are described using WSDL files. To generate the WSDL file:

1. In the Project Explorer, right-click the adapter instance of the Adapter Project that you want to deploy, select **Generate Adapter WSDL** from the shortcut menu.
2. In the Target Project dialog, select the SOA project that you want to use, click **OK**.

Two folders will be created once the process is done:

- `schema`: this folder contains the adapter XSD schema.
- *adapterConfigurationFilename*`_genresources`: this folder contains following supporting resources:
  - *adapterConfigurationFilename_number*`.dat`: contains information used by the Adapter container.
  - *adapterConfigurationFilename*`.adendpoints`: the adapter service endpoints file which contains information about the service endpoints supported by the adapter. The file is used later to select service types.
  - *adapterConfigurationFilename*`.substvar`: contains substitution variables that are imported from the adapter configuration file.
  - *adapterConfigurationFilename*`.wsdl`: the generated WSDL file which references the schema and `.dat` files contains the service definition endpoints.

The shared resource that defines the transport used to communicate with the adapter is created. If the adapter project referenced global variables, the shared resource will reference substitution variables defined in the *adapterConfigurationFilename*`.substvar` file.

After using a WSDL in a composite, ensure that you do not overwrite the WSDL.

Configuring Composites

In the ActiveMatrix platform, a *composite* exports a cohesive set of business functions as services. A composite contains of one or more components. A *component* contains a configured implementation, where the implementation provides the business functions.

The component configures the implementation with specific values for custom properties and by connecting component references to component services or composite references. The connections between references and services are represented by wires.

To configure a composite, follow the tasks below:

- Task A, Launch the Composite Editor, page 56
- Task B, Create a Shared Resource, page 56
- Task C, Add the Shared Resource to Composite, page 57
- Task D, Add Components, page 57
- Task E, Add Composite Service, page 58

For more information, refer to the TIBCO ActiveMatrix Service Grid documentation.

Task A Launch the Composite Editor

The Composite Editor is a graphical editor for developing composites.

In an SOA project, the composite file, *SOAProjectName*.`composite` is located at The `Composite` folder. Double-click the *SOA Project name*.`composite` file to launch the Composite Editor.

Task B Create a Shared Resource

In order for the ActiveMatrix service to be accessible from external client, a shared resource must be added.

1. From the File menu, select **New** > **TIBCO Shared resources**.
2. In the TIBCO Resource Wizard dialog, select wizard appropriate to the composite service type:
   - For JMS service, select **JMS**.
   - For SOAP service, select **HTTP Server**.

   Click **Next**.

3. Enter a name for the shared resource or accept the default. Specify the parent folder.
4. Click **Finish**.
5. Change the configuration in the Shared Resource Editor if necessary.

Task C Add the Shared Resource to Composite

To add the HTTP Server or JMS shared resource to the composite:

1. Click the canvas in the Composite Editor.
2. Click the **Shared Resource Profiles** tab in the Properties notebook.
3. In the Composite panel, click ... in the name column. A new shared resource profile is created. Enter a name or accept the default name.
4. In the type column, choose **HTTP** or **JMS**.
5. In the Target column, click ... to bring up the Select Shared Resource dialog.
6. Choose the shared resource and click **OK**.

Task D Add Components

A component contains a configured implementation. To add a component to a composite:

1. In the Composite Editor, drag the `IBMiAdapter` component from the Palette to the Components column. And enter a name for the component.
2. In the Properties notebook, click the **Services** tab. Click **Add**.
3. In the Resource Picker dialog, select an adapter endpoint from the left hand pane, and then click **Add** to add the adapter service to the component.
4. Click **OK**.

Figure 17 Add Resources to a Component

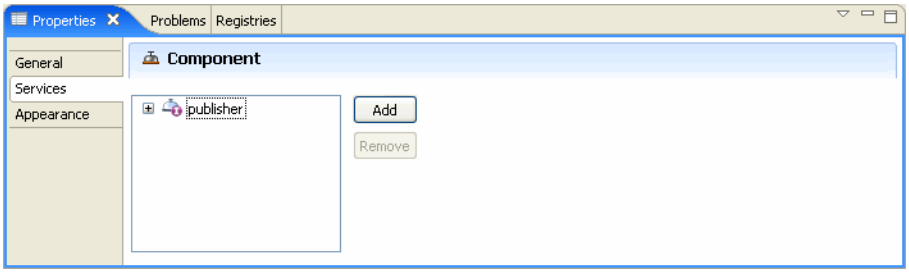


Table 11 lists the fields displayed in the General tab of the adapter component.

Table 11 Fields of the General Tab

Property Name	Description
Name	The adapter component name.
Description	The adapter component description.
Version	The adapter component version number.
Type	(Read-only) The adapter type.
SDK Application Name	(Read-only) The instance name of your adapter configuration. Gets populated after adding services to the component.
Configuration URI	(Read-only) The adapter configuration that the service uses. Gets populated after adding services to the component.
Repository URI	(Read-only) This file contains information used by the container to communicate with the adapter service engine. Gets populated after adding services to the component.
TRA File Name	The TRA file to use.

Task E Add Composite Service

A composite se rive represents a port type and it enables consumers outside the ActiveMatrix environment to access port types implemented within ActiveMatrix.

To add a composite service:

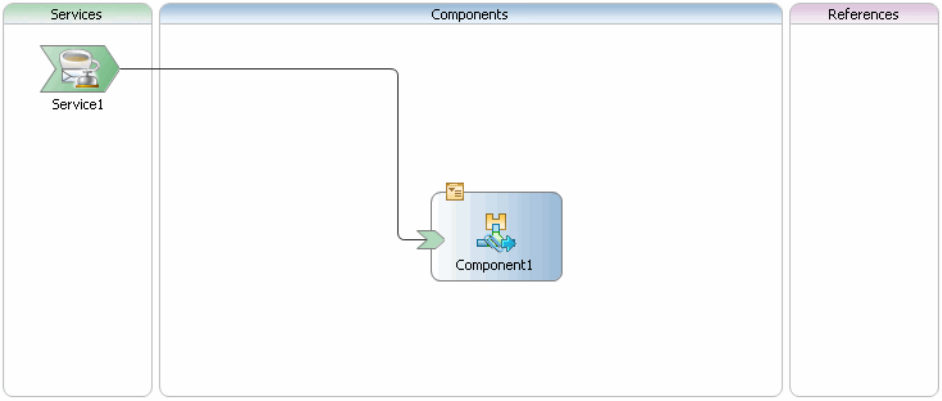
1. In the Composite Editor, drag a service from the Palette to the Services column.

   For TIBCO ActiveMatrix Adapter Service Engine for IBM i, only JMS and SOAP services are available.

2. Enter a name for the service.

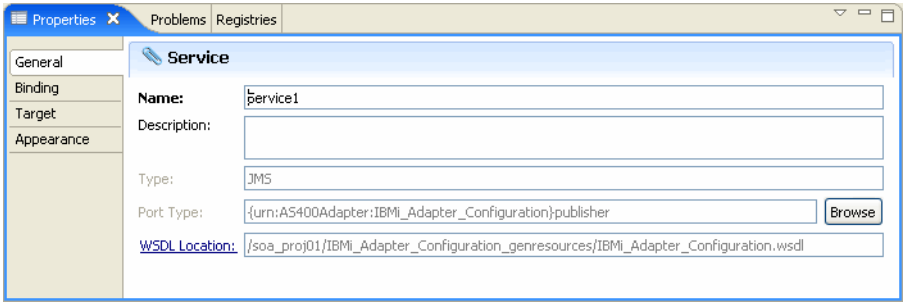
3. If you have added the component to the composite, click the **Target** tab of the Properties view, then select the target component to bind with the composite service.

Figure 18 Component and Composite Service Binding



Corresponding information is populated to the General tab and the Binding tab.

Figure 19 General Tab of Composite Service



4. Click the **Binding** tab in the Properties notebook to add the shared resource profile:
   - For JMS service, specify the shared resource profile in the Endpoint section.
   - For SOAP service, specify the transport type and shared resource profile in the Transport Configuration section.

   For the SOAP service, you also need to click the **Generate WSDL** button to generate another `.wsdl` file in the `Composites` folder of the SOA project. This `.wsdl` file will later be used by an external client to access the service within ActiveMatrix.

5. Save the configuration.

Creating Service Assembly and Service Assembly Archive

You need to package the adapter services that are created in the design phase into a service assembly before you can deploy them.

A service assembly contains service units and a descriptor that indicates the container into which each service unit is to be deployed into. The service assembly is then used to created a deployable archive, the service assembly archive.

Task A Create or Refresh a Service Assembly

- To create a service assembly:

1. In the Project Explorer, select the SOA project, expand the `Composites` folder.
2. Right-click the composite file, and then select **Service Assembly** from the shortcut menu.
3. Save the composite if prompted.

   The deployment package, *projectname*.`saf` is created in the `Deployment Packages` folder.

- To refresh a service assembly:

1. In the Project Explorer, select the SOA project, expand the `Deployment Packages` folder.
2. Right-click the service assembly file, and then select **Refresh Service Assembly** from the shortcut menu.

Task B Create a Deployable Archive

To create a deployable archive:

1. In the Project Explorer, select the SOA project, expand the `Deployment Packages` folder.
2. Right-click the `.saf` file, and then select **Build Archive** from the shortcut menu.
3. Save the `.saf` file if prompted.

   When the building process finished, you can find a `.zip` file in the `Deployment Packages` folder.

This service assembly archive can now be deployed and started using TIBCO ActiveMatrix Administrator. See Deploying the Service Assembly on page 65.

Chapter 7 Deploying the Service Assembly

This chapter describes how you can deploy a service assembly archive using TIBCO ActiveMatrix Administrator.

Topics

- Overview, page 62
- Using the EAR2SA Tool, page 63
- Deploying a Service Assembly Archive, page 64

Overview

The adapter service engine enables the adapter to run in the ActiveMatrix environment as a container and to provide services.

Adapter projects created using TIBCO Designer can also be packaged for deployment in the ActiveMatrix environment. Adapter projects can be designed to run in the following modes:

- Wired Mode—The adapter component is wired with other composite elements in the ActiveMatrix SOA project. Projects running in this mode interact with other components in the ActiveMatrix environment.

  Projects created using TIBCO Business Studio always run in the wired mode. See Creating Service Assembly and Service Assembly Archive on page 60 for more information about creating the service assembly archive.

- Unwired Mode—The adapter component in the ActiveMatrix SOA project is not wired with other composite elements and does not participate in the ActiveMatrix environment. Projects running in the unwired mode do not provide services to or consume services from other ActiveMatrix components. The adapter configurations merely run in the ActiveMatrix node but the deployments can be life-cycled using TIBCO ActiveMatrix Administrator.

  Projects imported using the `.ear` file always run in the unwired mode. See Using the EAR2SA Tool on page 63 for more information about creating service assembly archive from a `.ear` file.

Using the EAR2SA Tool

An adapter EAR file created using TIBCO Designer is first imported into TIBCO Business Studio.

To create a service assembly archive from an EAR file:

1. From the File menu, select **New > Project**.
2. In the New Project dialog, expand **TIBCO ActiveMatrix**, select **ActiveMatrix SOA Project**. Click **Next**.
3. Type a name in the Project name field and click **Finish**.
4. In the Project Explorer view, right-click the folder and select **Import...**.
5. Choose **General > File System**.
6. Click the **Browse** button to navigate to the directory where the `.ear` file is located.
7. Choose the `.ear` file and click **OK**.
8. In the Project Explorer view, right-click the imported `.ear` file and then select **Build Service Assembly Archive**.
9. The service assembly archive, *EAR_fileName*`.zip`, is created under the `Deployment Packages` folder.

The service assembly archive that is created can now be deployed.

The project will continue to use the encoding set in the Designer project.

Deploying a Service Assembly Archive

This section describes the tasks to configure and deploy a service assembly. These tasks are performed using TIBCO ActiveMatrix Administrator. Deploying a service assembly archive involves the following activities:

- Uploading a Service Assembly Archive, page 64
- Configuring the Service Assembly, page 64
- Deploying the Service Assembly, page 65
- Starting the Service Assembly, page 66

Before proceeding, ensure that the environment for TIBCO ActiveMatrix has been configured correctly. See TIBCO ActiveMatrix documentation for more information about how to configure the environment.

Uploading a Service Assembly Archive

The following steps describe how to upload the service assembly archive:

1. Select the **Deploy to an Environment** perspective.
2. Choose the Environment, and then click **Upload Service Assembly**.
3. In the **Name** field, enter a name for the service assembly.
4. Click **Browse** to select an archived ZIP file (the SA project) which was generated in TIBCO Business Studio.
5. Select the **Import Shared Resource Definitions** checkbox to create the shared resources definitions from the definitions in the service assembly archive.
6. Click **OK**.

The service assembly will appear in the Service Assemblies table.

Configuring the Service Assembly

To configure a service assembly, you need complete the following tasks.

Task A Configure the Service Unit

Specify the nodes on which the service units in the service assembly will be deployed.

1. In the Service Assemblies table, select the service assembly.

2. Click **Service Units** to view the service units in the selected service assembly.
3. For each service unit in the service assembly, do the following:
   a. Select a service unit in the Service Units table.
   b. In the Node Mapping tab, click **Edit**, select the nodes that you want to deploy the service unit to in the Available Nodes list, and then click the right arrow button. The nodes will appear in the Mapped Nodes list.
   c. Click **Save**.

The service unit properties are displayed in the following tabs:

- **Node Mapping** — you can map the nodes to service units. See *TIBCO ActiveMatrix Administration* for details.
- **Substitution Variables** — you can specify the value of the service unit's substitution variables. See *TIBCO ActiveMatrix Administration* for details.

Task B Start Node and Install Shared Resources

If an ActiveMatrix Service (JMS or SOAP) is included in the SA project, you need install the corresponding Shared Resource for the appropriate node.

1. In the web browser, select **Configure an Environment** in the Perspective drop-down list, and then select the appropriate Environment.
2. In the Nodes table, select the appropriate node. Click **Start**.
3. In the node detail panel, select **Shared Resources**.
4. Select the corresponding shared resource. Click **Edit**.
5. In the Enabled at Startup column, select the **Yes** radio button. Click **Save**.
6. On the top of the Shared Resources table, click **Install**.

Deploying the Service Assembly

A service assembly is ready to be deployed once the service assembly is uploaded and configured. Before proceed with deploying the service assembly, verify the following:

- All nodes required for deployment are running.
- All containers required for deployment are activated.
- All shared resources are installed.

To deploy a service assembly:

1. In the Service Assemblies table, select the service assembly.

2. Click **Deploy**.

Once the service assembly is deployed, the status changes to `Deployed`.

Starting the Service Assembly

After a service assembly has been successfully deployed, you can start it, and if required, stop it.

1. In the Service Assemblies table, select one or more service assemblies.
2. To start the selected service assemblies, click **Start**. The status changes to `Running`.
3. To stop the selected service assemblies, click **Stop**. The status changes to `Deployed`.

Chapter 8 Advanced Topics

This chapter explains advance topics of advanced features of TIBCO ActiveMatrix Adapter Service Engine for IBM i.

Topics

- Managing Logging, page 68
- Using the Adapter Service Engine with a Revision Control System, page 69
- Using Substitution Variables, page 70

Managing Logging

Trace messages are used to trace and troubleshoot an exception condition when an exception occurs at runtime.

You can configure the logging of trace messages using the TIBCO Common Logging framework by leveraging on the availability of Common Logging libraries in the ActiveMatrix environment. The Common Logging framework provides common messages and error codes that can be used across TIBCO products to log messages in a uniform manner.

To enable the common logging feature, the following properties are used:

- `application.ase.consoleoutput`
- `tibco.env.ADAPTER_ENABLE_CL`

The following table describes the usage of these properties.

Table 12 Properties for Common Logging Feature

Property Values	Behavior
`application.ase.consoleoutput = false` `tibco.env.ADAPTER_ENABLE_CL = false`	The common logging feature is disabled and the trace messages are written to the adapter log files.
`application.ase.consoleoutput = true` `tibco.env.ADAPTER_ENABLE_CL = false`	The common logging feature is disabled and the trace messages are displayed on the console. When the `application.ase.consoleoutput` property is set to `true`, any value assigned to the `tibco.env.ADAPTER_ENABLE_CL` property is ignored.
`application.ase.consoleoutput = false` `tibco.env.ADAPTER_ENABLE_CL = true`	The common logging feature is enabled.

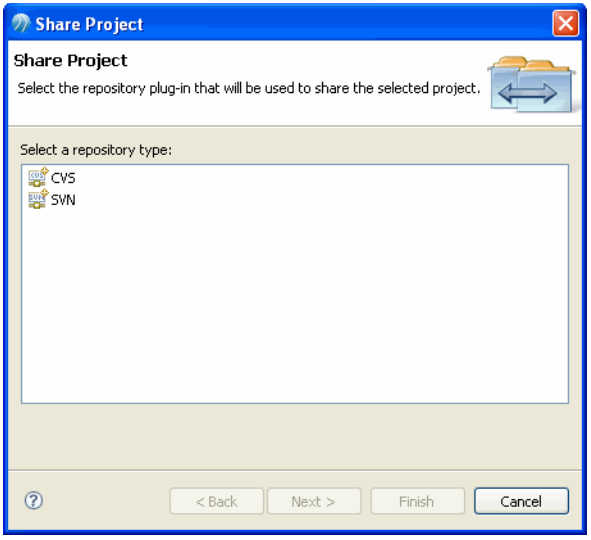
Using the Adapter Service Engine with a Revision Control System

TIBCO Business Studio supports revision control systems such as Microsoft Visual SourceSafe, TortoiseSVN, and Perforce.

To add a project to the revision control system:

1. Start TIBCO Business Studio.
2. In the Project Explorer, right-click the project that you want to add, select **Team > Share Project**.
3. In the Share Project dialog, select a repository type to share the project. Click **Next**.
4. Enter necessary parameters when prompted.

Figure 20 Add Files to the Revision Control System



Using Substitution Variables

The variable substitution mechanism can override substitution variables predefined in the project in a restricted manner. Predefined variables can be viewed and set in TIBCO Business Studio. Variables are specified as `%%VARNAME%%` and cannot contain any white space.

Variable substitution allows you to accomplish the following.

- Substitute string variables specified in the project at startup time.
- Locally define the value for a variable for a specific project. The local value takes precedence over any global value.
- Specify the value for a variable in a properties file. This overrides the project repository and values set in code, but not variables set on the command line.
- Enforce the pre-defined variables listed in Predefined Substitution Variables on page 72.

  Variables can be used anywhere in the configuration and will be replaced by the locally defined adapter instance.

Variable Specification

The adapter service engine can specify variables in the project during configuration using TIBCO Business Studio. It can also specify variables in a properties file. The properties file values overwrite values set in the project.

Specifying Substitution Variables Using TIBCO Business Studio

Substitution variables provide an easy way to set defaults for use throughout your project.

For example, you could assign the value 7600 to the predefined substitution variable `RvaPort`. And then use the variable in different sessions in the adapter service engine. To change the TIBCO Rendezvous daemon for the adapter service engine, globally set it to a different value or override it from the command line.

To use substitution variables in your project:

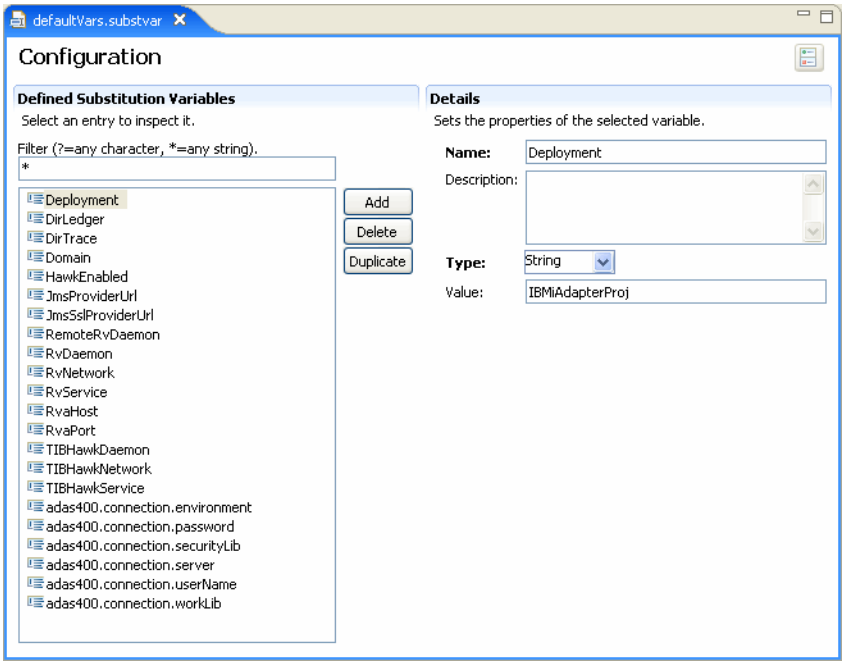
1. In the Project Explorer, expand **defaultVars**, double-click **defaultVars.substvar**.
2. All currently defined substitution variables are displayed in the editor view.

You can do the following:

— To assign or change a variable value, select the variable from the Defined Substitution Variables section, then go to the Details section, enter the new value in the Value field.

— To add a substitution variable, click **Add** in the Defined Substitution Variables section. A new substitution variable entry is added to the bottom of the list. Enter the variable name, type, and, optionally, the value and description.

Figure 21 Substitution Variables



3. To use the substitution variable in the fields of a resource, enter the variable name enclosed with `%%` on both sides or pick a substitution variable from the Binding Editor.

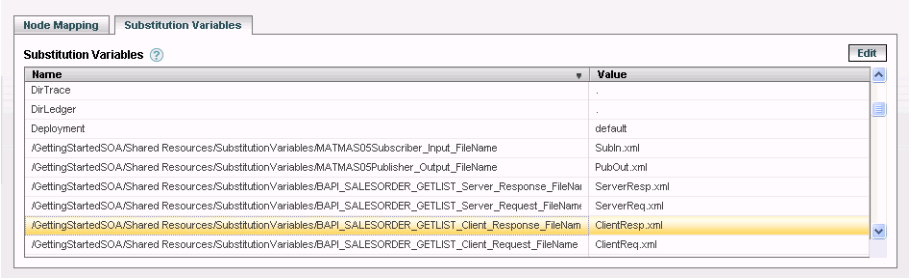
   When the project is deployed and the configured components are run, all occurrences of the substitution variable name are replaced with the substitution variable value (unless it was overridden in a way that had higher precedence). For example, `RvService` would be replaced with `7500`.

Changing Substitution Variables at Runtime

At runtime, you can update the values of the substitution variables using TIBCO ActiveMatrix Administrator.

Figure 22 shows the Substitution Variables editor in TIBCO ActiveMatrix Administrator, where you can specify runtime values for substitution variables.

Figure 22 Substitution Variables in TIBCO ActiveMatrix Administrator



A number of substitution variables are predefined. See Predefined Substitution Variables on page 72 for information. You can add definitions of any variables you need to the predefined variables.

Predefined Substitution Variables

Table 13 lists and explains the predefined substitution variables. Some of them are automatically used within the system when an adapter instance is configured.

Table 13 Predefined Substitution Variables

Variable	Description
`Deployment`	Defaults to the project name. This variable is used by the system to partially define the subject name defined for a service.
`DirLedger`	Used by the system when defining the path name of the TIBCO Rendezvous certified messaging ledger file. The default is root installation directory.
`DirTrace`	Used by the system to partially create the path name for log file used by the adapter. The default is the root installation directory.
`Domain`	The default value is `Domain`. This value can be any string value. This variable is used by the system to partially define the subject name defined for a service.

Table 13 Predefined Substitution Variables (Cont'd)

Variable	Description
`HawkEnabled`	Used by the system to indicate whether TIBCO Hawk is used to monitor the adapter. `True` indicates that a Hawk microagent is defined for the adapter. `False` indicates the microagent is not to be used.
`JmsProviderUrl`	Specifies where the EMS server is located. Setting this value mostly makes sense in early stages of a project, when only one EMS server is used.
`JmsSslProviderUrl`	Specifies where the EMS SSL daemon is located.
`RemoteRvDaemon`	Used by the system to identify the TIBCO Rendezvous routing daemon. See *TIBCO Rendezvous Administration* for details about specifying the routing daemon name.
`RvDaemon`	Used by the system to identify the TIBCO Rendezvous daemon parameter. The parameter instructs the transport object about how and where to find the Rendezvous daemon and establish communication. The default value is "tcp: `7500`", which is the default value used by the Rendezvous daemon. See *TIBCO Rendezvous Concepts* for details about specifying the daemon parameter.
`RvNetwork`	Used by the system to identify the TIBCO Rendezvous network parameter. Every network transport communicates with other transports over a single network interface. On computers with more than one network interface, the network parameter instructs the TIBCO Rendezvous daemon to use a particular network for all outbound messages from this transport. See *TIBCO Rendezvous Concepts* for details about specifying the network parameter.
`RvService`	Used by the system to identify the TIBCO Rendezvous service parameter. The Rendezvous daemon divides the network into logical partitions. Each transport communicates on a single service; a transport can communicate only with other transports on the same service. The default value is `7500`. You can change the default value depending on your service requirement. See *TIBCO Rendezvous Concepts* for details about specifying the service parameter.
`RvaHost`	Used by the system to identify the computer on which the TIBCO Rendezvous agent runs. See *TIBCO Rendezvous Administration* for details about specifying the rva parameters.
`RvaPort`	Used by the system to identify the TIBCO Rendezvous agent TCP port where the agent listens for client connection requests. See *TIBCO Rendezvous Administration* for details about specifying the rva parameters.
`TIBHawkDaemon`	Used by the system to identify the TIBCO Hawk daemon parameter. See the *TIBCO Hawk Installation and Configuration* manual for details about this parameter.
`TIBHawkNetwork`	Used by the system to identify the TIBCO Hawk network parameter. See the *TIBCO Hawk Installation and Configuration* manual for details about this parameter.

Table 13 Predefined Substitution Variables (Cont'd)

Variable	Description
`TIBHawkService`	Used by the system to identify the TIBCO service parameter. See the *TIBCO Hawk Installation and Configuration* manual for details about this parameter.
adas400.connection.environment	Identifies the connection options to the IBM System i machine. See Configuring an IBM i Connection on page 27 for more information.
adas400.connection.password	
adas400.connection.securityLib	
adas400.connection.server	
adas400.connection.userName	
adas400.connection.worklib	

Chapter 9 Monitoring the Adapter Using TIBCO Hawk

Read this chapter if you have installed TIBCO Hawk and want to use TIBCO Hawk microagents through the Configuration tab of an adapter instance. TIBCO Hawk can be used in addition to the standard trace message logging methods described in Logging Sinks Tab on page 49.

In the TIBCO ActiveMatrix BusinessWorks environment, the adapter is monitored using TIBCO Administrator. See TIBCO ActiveMatrix BusinessWorks documentation for more information.

Topics

- Overview, page 76
- Starting TIBCO Hawk Software, page 77
- The Auto-Discovery Process, page 78
- Invoking Microagent Methods, page 79
- Available Microagents, page 82

Overview

TIBCO Hawk is a tool for enterprise-wide monitoring and managing of all distributed applications and systems. System administrators can use it to monitor adapters in a wide area network of any size.

TIBCO Hawk can be configured to monitor system and adapter parameters and to take actions when predefined conditions occur. These actions include:

- sending alarms that are graphically displayed in the TIBCO Hawk display
- sending e-mail
- paging
- running executable
- modifying the behavior of a managed adapter

Unlike other monitoring applications, TIBCO Hawk relies on a purely distributed intelligent agent architecture using publish or subscribe to distribute alerts.

TIBCO Hawk uses TIBCO Rendezvous for all messaging and thus gains the benefits and scalability from the TIBCO Rendezvous features of publish or subscribe, subject name addressing, interest-based routing, and reliable multicast.

TIBCO Hawk is a purely event-based system that uses alerts. The agents are configured with rules that instruct them on everything from what and how to monitor to what actions to take when problems are discovered. Thus, the workload is fully distributed throughout the enterprise. Every agent is autonomous in that it does not depend on other components to perform its functions.

The TIBCO Hawk Enterprise Monitor consists of following components:

Display GUI front end that displays alarms and provides editors to create rule bases, create tests, view messages, and invoke microagents to request information or initiate an action.

Agents Intelligent processes that perform monitoring and take actions as defined in rules.

Rulebases Rules that are loaded by agents to determine agent behavior.

Application Management Interface (AMI) Manages network applications via TIBCO Rendezvous and supports communication between a network application and monitoring TIBCO Hawk agents, including the ability to examine application variables, invoke methods, and monitor system performance.

Microagents Feed information back to TIBCO Hawk and expose action methods to rulebases.

Starting TIBCO Hawk Software

The TIBCO Hawk agent can be configured to start automatically during the system boot cycle. See the *TIBCO Hawk Installation and Configuration* for instructions on how to start TIBCO Hawk.

The *TIBCO Hawk Administrator's Guide* explains how to start the TIBCO Hawk Display.

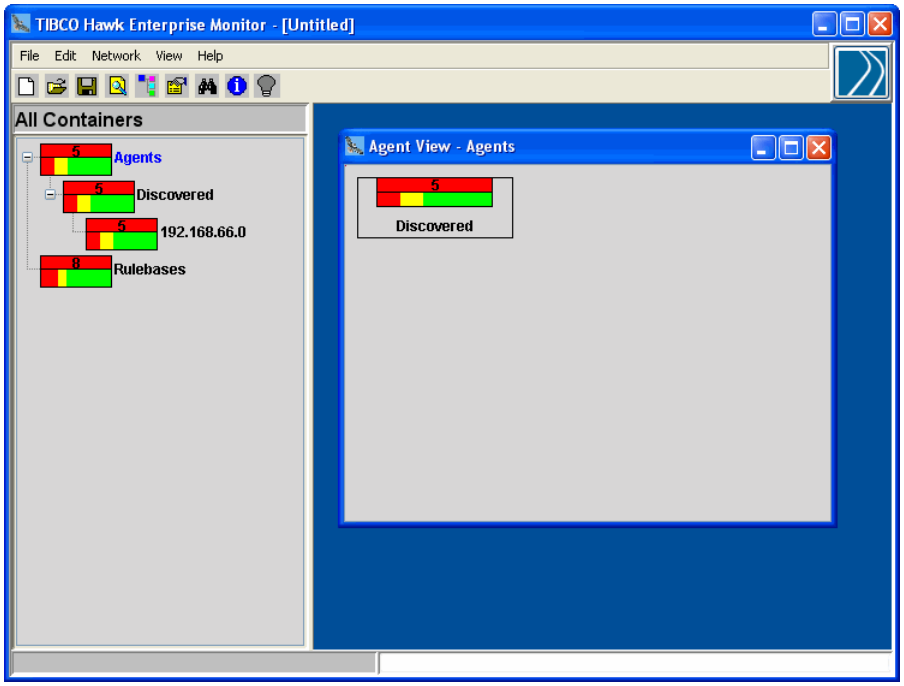
The guides are included in the TIBCO Hawk software installation area.

The Auto-Discovery Process

Start an instance of TIBCO Hawk Display, which continually discovers machines running TIBCO Hawk Agents on your network. Container icons are created for each agent, and arranged hierarchically in clusters. By default, agent icons are clustered according to subnets.

At first, the Agents container is empty. Its counter displays a value of zero and the Discovered counter is also at zero. Both icons are initially green to show that no alerts or warning messages are in effect. As agents are discovered, the counters increase to reflect the current number.

Figure 23 TIBCO Hawk Enterprise Monitor



Monitored network nodes are arranged in a hierarchical tree of containers. Clicking a container in the left-hand panel displays nested items on the right-hand panel.

Icon colors change to reflect the highest level of alert found on discovered agents. See *TIBCO Hawk Administrator's Guide* for more information about icon elements and characteristics.

Invoking Microagent Methods

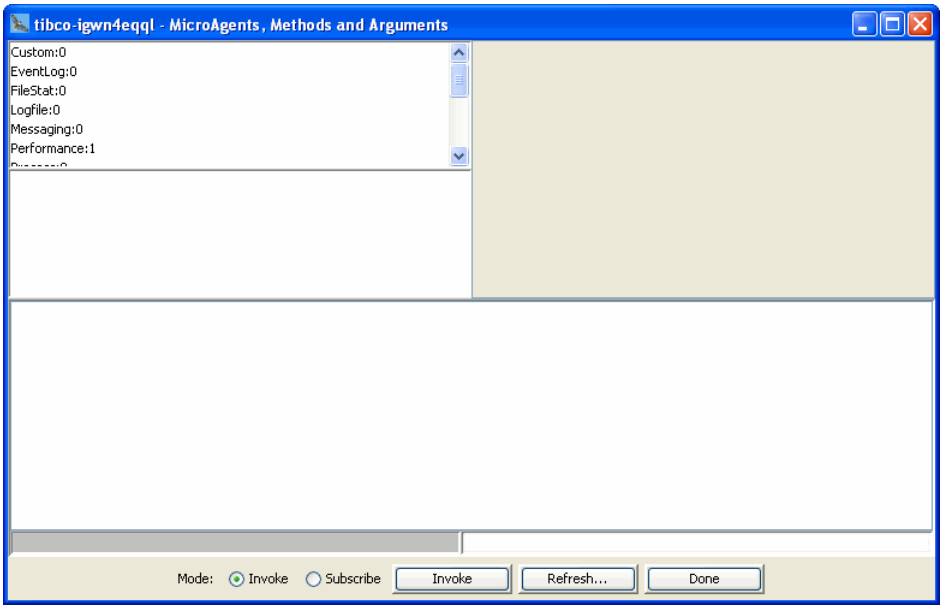
A set of default microagents is loaded when a TIBCO Hawk Agent is started. When you install and start the adapter, its microagents are dynamically added to the local agent. To invoke a microagent method on a TIBCO Hawk Agent:

1. In TIBCO Hawk Display, right-click the **Agent** icon and select **Get Microagents**.

   If TIBCO Hawk security is implemented on your system and you do not have access to microagents on this agent, an error dialog appears. Select another agent, or contact your system administrator to obtain access.

   The Microagents, Methods and Arguments dialog appears. The panel on the upper left lists microagents you can access on the current agent.

Figure 24 Microagents, Methods and Arguments Dialog



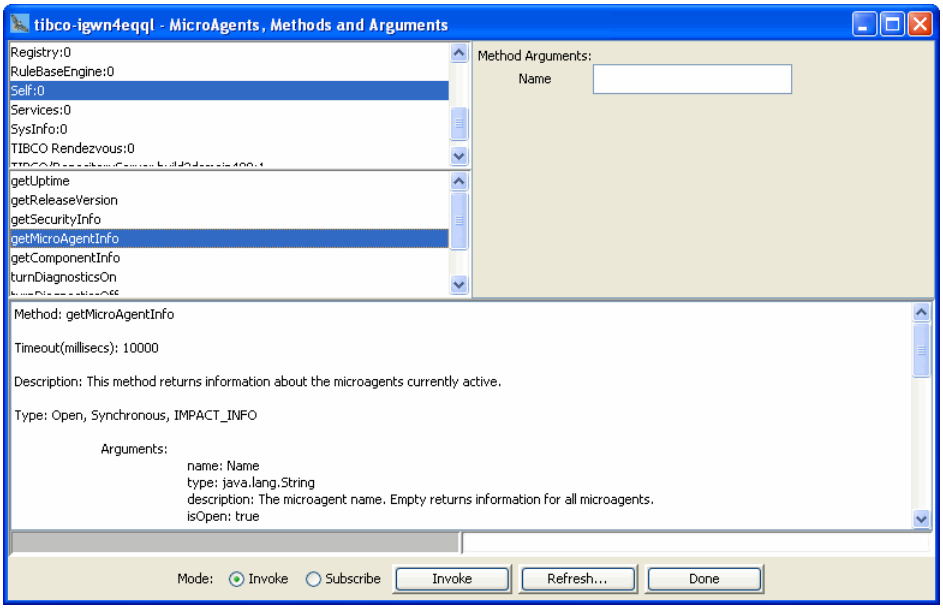
   This dialog has two modes that are controled by the radio buttons at the bottom:

   — `Invoke`: Invoking a method immediately returns a single set of current results.

   — `Subscribe`: Subscribing provides updates of current results at regular intervals.

2. Click a microagent name, such as **Self**, to display a list of associated methods and text descriptions in the panels below.
3. Click the name of the method to invoke, such as **getMicroAgentInfo**.

Figure 25 Select Microagent and Method

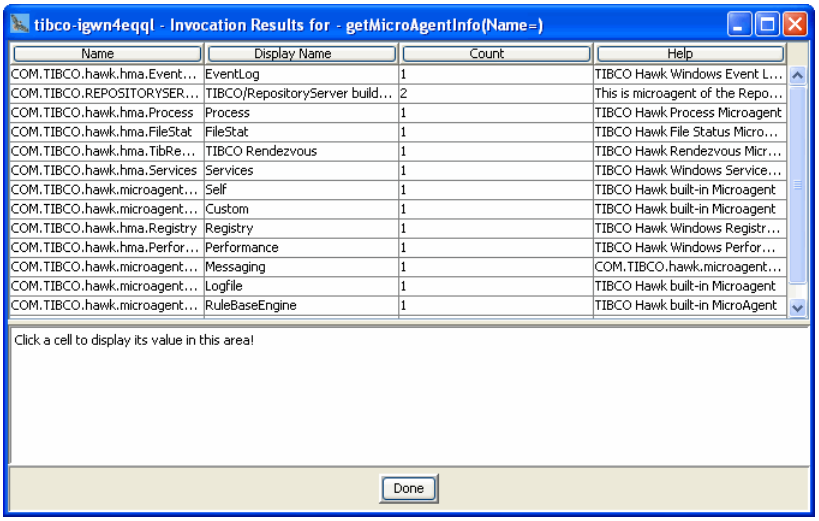


If the method accepts arguments, fields for each argument appear in the upper right panel. Detailed help text appears in the lower panel.

4. Specify any arguments for the method invocation.
5. Ensure that the **Invoke** option button is selected.
6. Click **Invoke** to invoke the selected method.

   The Invocation Results dialog displays the results returned by the method.

Figure 26 Invocation Result Dialog



7. Click **Done** to close the dialog.

These steps describe how to interactively invoke a microagent method and receive a single set of results in TIBCO Hawk Display. You can also use a microagent method as the data source of a TIBCO Hawk rule. Rules automatically receive method results, apply tests to evaluate them, then take action if necessary. For more information on building TIBCO Hawk rules and rule bases, see *TIBCO Hawk Administrator's Guide*.

Available Microagents

Each adapter has three microagents, a standard TIBCO Hawk microagent named `COM.TIBCO.ADAPTER.`*xyz* where *xyz* is the adapter configuration name, a custom microagent, and a class microagent (for PAM compliance).

The microagents provide:

- Business level statistics that report the progress of the adapter as it interacts with the vendor application.

  For example, in a database adapter such statistics might indicate whether objects were successfully or unsuccessfully inserted, updated, or deleted in the database.

- Queries that return information about the state of the adapter.

  This can be an important tool for seeing the internal of an adapter and debugging it if something appears wrong. For example, methods can return information about threads, internal queues, or connections to the target system. Using these methods, one might be able to identify certain bottlenecks or gauge how successfully an adapter is scaling with respect to the current environment.

- Updates of the adapter runtime parameters.

  This includes retrieving the current runtime parameters and setting new runtime parameters without restarting the adapter. An example of this is getting and setting the polling interval. Updating a runtime parameter through the Hawk microagent only affects the setting of the instance that is running. It does not make a permanent change of the setting in either the repository or the `.tra` file.

By default, only standard and class microagents are available at runtime. You can disallow adding custom methods to the class microagent when deploying the adapter by changing the `addCustomHawkMethodsToClassMAgent` property value in the adapter's property file.

Table 14 lists all the standard methods available for the adapter.

Table 14 Microagent Methods

Method	Description
`activateTraceRole()`	Activates a mapping of a role to a sink at runtime.
`deactivateTraceRole()`	Deactivates a mapping of a roles to sinks at runtime.

Table 14 Microagent Methods (Cont'd)

Method	Description
`getActivityStatistics()`	Returns the total number of objects processed for all the schemas.
`getActivityStatisticsByOperation()`	Returns the total number of objects processed for all the schemas by each service that is associated with a specified operation.
`getActivityStatisticsByService()`	Returns information about the services implemented by this adapter.
`getAdapterServiceInformation()`	Returns information about the services implemented by this adapter.
`getComponents()`	Returns information about the publisher, subscriber and IODescriptor.
`getConfig()`	Returns basic configuration information. More specific information is accessed by the more specific methods.
`getConfigProperties()`	Returns all attributes and elements for the given repository object.
getConnectionStatistics()	Returns the state and statistics for all the current connections used by the adapter.
`getHostInformation()`	Returns standard and extended application information.
`getQueueStatistics()`	Returns the current count of elements in any internal queue used by the adapter.
`getRvConfig()`	Returns information about all TIBCO Rendezvous sessions defined.
`getStatus()`	Returns general status information, such as the number of TIBCO Rendezvous messages received and published, the number of errors since the last call, the PID of the application, and more.
`getThreadStatistics()`	Returns the operation counts of the current threads.
`getTraceSinks()`	Returns information about sinks to which traces currently go.
`getVersion()`	Returns the configuration ID, application name, version, and date for this adapter instance.

Table 14 Microagent Methods (Cont'd)

Method	Description
`_onUnsolictedMsg()`	Displays alert messages sent to the current adapter.
`preRegisterListener()`	Preregisters an anticipated listener.
`resetActivityStatistics()`	Resets all the counts for the activity statistics.
`resetConnectionStatistics()`	Resets all the counts for the connection statistics.
`resetThreadStatistics()`	Resets all the counts for the thread statistics.
`reviewLedger()`	Returns information retrieved from the ledger file of a certified messaging session for a publisher adapter.
`setTraceSinks()`	Adds a role or changes the file limit of a previously specified sink.
`stopApplicationInstance()`	Stops the running adapter instance.
`unRegisterListener()`	Unregisters a currently preregistered listener.

activateTraceRole()

Purpose

Activates a mapping of a role to a sink at runtime. This replaces the now-deprecated `setTraceSink()` TIBCO Hawk method.

Parameters

Table 15 Input parameters of activateTraceRole()

Input Parameters	Type	Description
roleName	string	Name of the role to activate.
sinkName	string	Name of the sink for which to activate the role.

deactivateTraceRole()

Purpose Deactivates a mapping of roles to sinks at runtime.

Parameters

Table 16 Input parameters of deactivateTraceRole()

Input Parameters	Type	Description
roleName	string	Name of the role to deactivate.
sinkName	string	Name of the sink for which to deactivate the role.

getActivityStatistics()

Purpose

Returns the total number of objects processed for all the schemas, based on the request type. Also, returns the number of success and error objects.

Parameters

Table 17 Input parameter of getActivityStatistics()

Input Parameter	Type	Description
GetSubTotalBy	string	Indicates how to group the subtotals, by `Service` or `Operation`.

Returns

Table 18 Returns of getActivityStatistics()

Returns	Type	Description
Name	string	Service name or `All Services` which represents the final tally of all the services.
Total	integer	Total number of objects processed including both success and failures.
Success	integer	Total number of objects successfully processed.
Failure	integer	Total number of objects that caused an error during processing.
MeasurementInterval	integer	Displays the time (in seconds) since last time the adapter was reset, or if never reset, since the adapter started.

getActivityStatisticsByOperation()

Purpose Returns statistics about one operation.

Parameters

Table 19 Input parameter of getActivityStatisticsByOperation()

Input Parameter	Type	Description
Operation	string	Name of the operation.

Returns

Table 20 Returns of getActivityStatisticsByOperation()

Returns	Type	Description
Operation	string	Name of the operation.
Service Name	string	Name of the service.
Total	integer	Total number of objects processed, both success and failures.
Success	integer	Total number of objects successfully processed.
Failure	integer	Total number of objects that caused an error during processing.
MeasurementInterval	integer	Displays the time (in seconds) since last time the adapter was reset, or if never reset, since the adapter started.
LineIndex	string	Concatenated string of `Service Name` and `Operation` separated by a comma.

getActivityStatisticsByService()

Purpose

Returns statistics about the data handled by a given adapter service or all adapter services since the time the adapter was started.

Parameters

Table 21 Input parameter of getActivityStatisticsByService()

Input parameter	Type	Description
Service Name	string	Name of service to get the statistics for. If no service name is given, performance statistics for all services is returned.

Returns

Table 22 Returns of getActivityStatisticsByService()

Returns	Type	Description
Service Name	string	Service name.
Schema Name	string	Name of top level schema processed by this service.
Operation	string	Type of operation this service provides.
Total	integer	Total number of objects processed, both success and failures.
Success	integer	Total number of objects successfully processed.
Failure	integer	Total number of objects that caused an error during processing.
MeasurementInterval	integer	Displays the time (in seconds) since last time the adapter was reset, or if never reset, since the adapter started.
LineIndex	string	Concatenated string of `Service Name` and `Operation` separated by a comma.

getAdapterServiceInformation()

Purpose

Returns information about the services implemented by this adapter. The information is a summary of available adapter services.

Parameters

Table 23 Input parameter of getAdapterServiceInformation()

Input Parameter	Type	Description
ServiceName	string	Name of the service from which to get information. Default is ALL.

Returns

Table 24 Returns of getAdapterServiceInformation()

Returns	Type	Description
Line	integer	Sequential row number.
ServiceName	string	Name of the service as defined at design-time.
EndpointName	string	Name of the endpoint used for this service.
Type	string	Type of the endpoint, for example, publisher or subscriber.
Quality of Service	string	Quality of service for the endpoint. For example, RVCM or JMS Persistent.
Subject	string	Subject defined for this endpoint.
Class	string	Class associated with the endpoint.
Number of Messages	integer	Number of messages processed for this endpoint.

getComponents()

Purpose

Returns information about the currently active TIBCO Hawk components such as publishers, subscribers, or timers.

Parameters

Table 25 Input parameters of getComponents()

Input Parameters	Type	Description
Component Name	string	Name of the component. If no value is provided, all components are displayed.
Component Type	string	Any of `Publisher`, `Subscriber`, `Timer`, or `IODescriptor`. The default value is `All`.

Returns

Table 26 Returns of getComponents()

Returns	Type	Description
Instance ID	string	Name of this adapter instance as defined at design time.
Adapter Name	string	Name of the adapter.
Component Name	string	Name of the component.
Component Type	string	The name of the TIBCO Adapter SDK class for this component, such as `MPublisher`, `MSubscriber`, or `MIODescriptorSource`. For more information about the class, see your TIBCO Adapter SDK documentation.
Session Name	string	Name of the session.
Description	string	Information about this component, for example, time interval, signal type, validating the publisher or subscriber.

getConfig()

Purpose Retrieves generic configuration information. More specific configuration information is accessed through separate methods.

Returns

Table 27 Returns of getConfig()

Returns	Type	Description
Instance ID	string	Configuration ID of this adapter.
Adapter Name	string	Name of the adapter.
Repository Connection	string	URL of the repository used for adapter instance.
Configuration URL	string	Location of the adapter project; either a file name or configuration URL.
Command	string	Command line arguments used to start the adapter.

getConfigProperties()

Purpose

Returns all attributes and elements for the given repository object.

Parameters

Table 28 Input parameter of getConfigProperties()

Input Parameter	Type	Description
Property	string	Name of the property for which elements (tags) and attributes are desired. For example, `agentone/startup`. If no value is given, all properties are returned.

Returns

Table 29 Returns of getConfigProperties()

Returns	Type	Description
Element Name	string	Repository directory for the property.
Attribute Name	string	Name of the repository object attribute.
Attribute Value	string	Value of the repository object attribute.
Line	integer	Line number in which this property is defined in the project file.

getConnectionStatistics()

Purpose Returns the state and statistics for all the current connections used by the adapter.

Returns

Table 30 Returns of getConnectionStatistics()

Returns	Type	Description
Connection ID	string	Unique identification of a particular connection.
Connection Type	string	Type or key that will match this connection to a thread or queue.
State	string	Current state: CONNECTED or DISCONNECTED.
NumRetries	integer	Total number of times this connection had to be reestablished.
TotalNumOperations	integer	Total number of operations processed by this connection since the adapter started.
CurrentNumOperations	integer	Total number of operations processed by this connection since the last reconnection.
NumLostConnections	integer	Total amount of time that this connection has been lost.
MeasurementInterval	integer	Displays the time (in seconds) since last time the adapter was reset, or if never reset, since the adapter started.

getHostInformation()

Purpose Returns standard and extended application information set.

Returns

Table 31 Returns of getHostInformation()

Returns	Type	Description
Name	string	Name of the property.
Value	string	Value of the property.

getQueueStatistics()

Purpose

Returns the current count of elements in any internal queue used by the adapter. This includes the TIBCO Rendezvous event queues automatically spawned by Rendezvous for each adapter.

Returns

Table 32 Returns of getQueueStatistics()

Returns	Type	Description
QueueID	string	Unique identification of a particular queue.
QueueType	string	Type or key that will match this queue to a thread or connection.
QueueCount	integer	Current number of elements in the queue.
MaxQueueSize	integer	Maximum number of elements in the queue.
MeasurementInterval	integer	Displays the time (in seconds) since last time the adapter was reset, or if never reset, since the adapter started.

getRvConfig()

Purpose

Returns information about the TIBCO Rendezvous session defined by this adapter.

Parameters

Table 33 Input parameter of getRvConfig()

Input Parameter	Type	Description
Session Name	string	Name of the TIBCO Rendezvous session for which configuration is required. If not given, information about all currently defined sessions is returned. The default is `all`.

Returns

Table 34 Returns of getRvConfig()

Returns	Type	Description
Instance ID	string	The configuration ID of this adapter.
Adapter Name	string	Name of the adapter.
Session Name	string	Name of the session.
Service	string	Service parameter for this session.
Daemon	string	Daemon parameter for this session.
Network	string	Network parameter for this session.
Synchronous	boolean	Returns `1` if this is a synchronous session, `0` otherwise.
Session Type	string	Type of session; one of `M_RV`, `M_RVCM`, or `M_RVCMQ`.
Certified Name	string	Name of this certified session.
Ledger File	string	Ledger file for this certified messaging session. Returns the empty string for sessions that are not certified messaging sessions.
CM Timeout	string	Timeout for this certified messaging session. Returns the empty string for sessions that are not certified messaging sessions.

getStatus()

Purpose

Retrieves basic status information about the adapter.

This information is fairly limited. Additional methods are provided in getConfig() on page 92 and getRvConfig() on page 97.

Returns

Table 35 Returns of getStatus()

Returns	Type	Description
Instance ID	string	Configuration ID for this adapter instance.
Adapter Name	string	Name of the adapter.
Uptime	integer	Number of seconds since startup.
Messages Received	integer	Number of TIBCO Rendezvous messages received.
Messages Sent	integer	Number of TIBCO Rendezvous messages published.
New Errors	integer	Number of errors since the last call to this method.
Total Errors	integer	Total number of errors since startup.
Process ID	integer	Process ID of the application.
Host	string	Name of host machine on which this adapter is running.

getThreadStatistics()

Purpose Returns the operation counts of the current threads.

Returns

Table 36 Returns of getThreadStatistics()

Returns	Type	Description
ThreadID	string	Unique identification of a particular thread.
ThreadType	string	Type that tells what part of the adapter this thread belongs to. Valid types include "Publisher", "Subscriber", "RPC", or "Connection".
TaskType	string	One-word description of the tasks this thread processes.
TaskCount	integer	Number of tasks processed by this thread.
MeasurementInterval	integer	Displays the time (in seconds) since last time the adapter was reset, or if never reset, since the adapter started.

getTraceSinks()

Purpose

Returns information about sinks to which traces currently go.

Parameters

Table 37 Input parameters of getTraceSinks()

Input Parameters	Type	Description
Sink Name	string	Name of the sink for which you need information. If no name is specified, information about all sinks is returned. Default is `all`.
Role Name	string	Name of the role for which you need information for the specified sink or sinks. Default is `all`.

Returns

Table 38 Returns of getTraceSinks()

Returns	Type	Description
Instance ID	string	Name of this adapter instance as a string.
Adapter Name	string	Name of the application for this sink.
Sink Name	string	Name of the sink.
Sink Type	string	Type of this sink. One of fileSink, rvSink, hawkSink, stderrSink.
Roles	string	Roles this sink supports, as a string. For example “warning, error, debug”.

getVersion()

Purpose Retrieves version information for the current application. Two lines may be returned, one for the TIBCO Adapter SDK, one for the adapter.

Returns

Table 39 Returns of getVersion()

Returns	Type	Description
Instance ID	string	The configuration ID as a string, for example, `SDK`.
Adapter Name	string	Name of the adapter as a string, for example, `agentone`.
Version	string	Version number as a string, for example, `1.1`.

_onUnsolictedMsg()

Purpose Displays all alert messages sent from the adapter or an error if not successful.

preRegisterListener()

Purpose

Preregisters an anticipated listener. Some sending applications can anticipate requests for certified delivery even before the listening applications start running. In such situations, the sender can preregister listeners, so TIBCO Rendezvous software begins storing outbound messages in the sender's ledger. If the listening correspondent requires old messages, it receives the backlogged messages when it requests certified delivery.

Parameters

Table 40 Input parameters of preRegisterListener()

Input Parameters	Type	Description
Session Name	string	Name of the session that anticipates the listener.
Publisher Name	string	Name of the component for which the listener should be preregistered.
Listener Session Name	string	Name of the listener to preregister.

Returns

`OK` if the listener was preregistered successfully, `false` otherwise.

resetActivityStatistics()

Purpose Resets all the counts for the activity statistics.

resetConnectionStatistics()

Purpose Resets all the counts for the connection statistics.

resetThreadStatistics()

Purpose Resets all the counts for the thread statistics.

reviewLedger()

Purpose

Returns information retrieved from the ledger file of a TIBCO Rendezvous certified messaging session.

Before invoking this method, ensure that the certified messaging publisher adapter has established a certified delivery agreement with its subscriber agents.

Parameters

Table 41 Input parameters of reviewLedger()

Input Parameters	Type	Description
Session Name	string	Name of the TIBCO Rendezvous session for which ledger information is desired (default is `all`).
Subject	string	Name of the subject for which ledger information is desired.

Returns

Table 42 Returns of reviewLedger()

Return	Type	Description
Session Name	string	Name of the TIBCO Rendezvous CM session to which this information applies.
Subject	string	Subject name for this session.
Last Sent Message	integer	Sequence number of the most recently sent message with this subject name.
Total Messages	string	Total number of pending messages with this subject name.
Total Size	integer	Total storage (in bytes) occupied by all pending messages with this subject name. If the ledger contains ten messages with this subject name, then this field sums the storage space over all of them.
Listener Session Name	string	Within each listener submessage, the Listener Session Name field contains the name of the delivery-tracking listener session.

Table 42 Returns of reviewLedger() (Cont'd)

Return	Type	Description
Last Confirmed	string	Within each listener submessage, the Last Confirmed field contains the sequence number of the last message for which this listener session confirmed delivery.
Line	integer	Row number in ledger file.
UnacknowledgedMessages	integer	Number of RVCM messages pending for this listener. The value is computed by subtracting the last sent sequence number from the last acknowledged sequence number.

setTraceSinks()

Purpose Adds a role or changes the file limit of a previously specified sink.

Parameters

Table 43 Input parameters of setTraceSinks()

Input Parameters	Type	Description
Sink Name	string	Name of the sink for which you want to add a role or change the file limit.
Role Name	string	Name of the role you want to add to this sink (warning, error, debug, or user defined). Default is all.
File Size	integer	Maximum file size for this sink. This parameter is ignored if the sink specified by sinkName is not a file sink.

Returns OK if successful or an error if not successful.

stopApplicationInstance()

Purpose Stops the specified adapter by calling the internal `stop()` method.

Returns `OK` if successful or an error if not successful.

unRegisterListener()

Purpose Unregisters a currently preregistered listener.

Parameters

Table 44 Input parameters of unRegisterListener()

Input Parameters	Type	Description
Session Name	string	Name of the session that anticipates the listener.
Publisher Name	string	Name of the component to which the listener is preregistered.
Listener Session Name	string	Name of the listener to be unregistered.

Returns `True` if the listener was unregistered successfully, `false` otherwise.

Appendix A Status Messages

This appendix explains the trace messages that are logged to a location specified at configuration time.

Topics

- Overview, page 114
- Trace Message Fields, page 116
- Status Messages, page 118

Overview

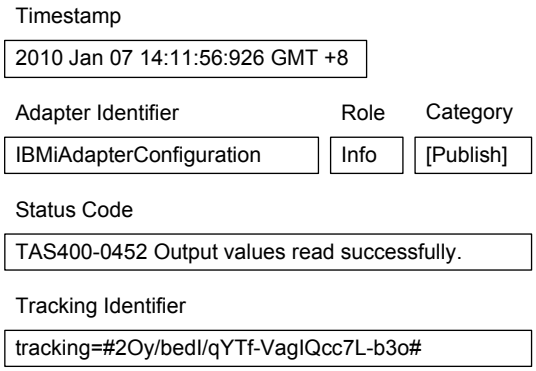
Trace messages provide information about adapter activities. The messages are logged to the console where the runtime adapter is started and to a log file. Trace messages can also be redirected to the TIBCO Hawk Display application, or sent to other applications using the TIBCO Rendezvous transport.

Each trace message can include the following fields:

```
<Timestamp> <Adapter Identifier> <Role> <Category> <Status Code>
<Tracking Identifier>
```

The above fields are explained in Trace Message Fields on page 116. Figure 27 shows an example trace message and calls out the fields.

Figure 27 Trace Message



Example Trace Messages

The following trace messages are written during a session in which the Publication Service monitors an outbound data queue and publishes messages placed in the queu.

The first message indicates that TIBCO ActiveMatrix Adapter Service Engine for IBM i has started. The timestamp indicates when the adapter started, and the role indicates that the trace message is informational, which means the activity is normal for the adapter. The category is identified, and the corresponding status code is displayed. The status code indicates that the adapter started successfully.

```
2010 Jan 07 14:06:20:415 GMT +8 IBMiAdapterConfiguration Info
[Adapter] TIS-00001 "Adapter AS400Adapter started successfully"
```

The next set of trace messages indicates that the adapter obtained connection from the pool and read output values successfully. The tracking identifier included in the trace message uniquely identifies the message.

```
2010 Jan 07 14:11:54:831 GMT +8 IBMiAdapterConfiguration Info
[OutMsg] TAS400-0351 Received message from data queue ADAS400PUB in
library ADAS400.

2010 Jan 07 14:11:54:924 GMT +8 IBMiAdapterConfiguration Info
[Publish] TAS400-0451 Publisher CustomerMessage_Publisher received
message. tracking=#2Oy/bedI/qYTf-VagIQcc7L-b3o#

2010 Jan 07 14:11:56:895 GMT +8 IBMiAdapterConfiguration Info
[Adapter] TIS-30028 "Obtained connection from pool  "

2010 Jan 07 14:11:56:926 GMT +8 IBMiAdapterConfiguration Info
[Publish] TAS400-0452 Output values read successfully.
tracking=#2Oy/bedI/qYTf-VagIQcc7L-b3o#

2010 Jan 07 14:11:56:973 GMT +8 IBMiAdapterConfiguration Info
[Publish] TAS400-0453 Publisher CustomerMessage_Publisher
published message successfully.
tracking=#2Oy/bedI/qYTf-VagIQcc7L-b3o#
```

The final trace message indicates that the Publication Service published the message successfully.

Trace Message Fields

Table 45 lists the fields in trace messages.

Table 45 Tracing Fields

Field Name	Description
Timestamp	Timestamp of occurrence. For example, `2010 Jan 07 14:06:20:415 GMT +8`.
Adapter Identifier	Name of the adapter that wrote the trace message. This is a combination of the adapter acronym and adapter configuration name.
Role	A role can be: **Info** Indicates normal adapter operation. No action is necessary. A tracing message tagged with Info indicates that a significant processing step was reached and has been logged for tracking or auditing purposes. Only info messages preceding a tracking identifier are considered significant steps. **Warn** Indicates an abnormal condition. Processing will continue, but special attention from an administrator is recommended. **Error** An unrecoverable error occurred. Depending on the error severity, the adapter may continue with the next operation or stop altogether. **Debug** A developer-defined tracing message. In normal operating conditions, debug messages should not display. When configuring the adapter you define what roles should or should not be logged. For example, you may decide not to log Info roles to improve performance.

Table 45 Tracing Fields (Cont'd)

Field Name	Description
Category	One of the following: **Adapter** The adapter is processing an event. **Application** The adapter is interacting with the IBM System i. **Configuration** The adapter is reading configuration information. **Database** The adapter is interacting with a database. **DTA** (Design-time adapter) The trace message is from the DTA. **Metadata** The adapter is retrieving metadata from the IBM System i. **Palette** The adapter is interacting with the palette. **Publisher Service** Publication Service reports the trace message. **Request-Response Client Service** Request-Response Invocation Service reports the trace message. **Request-Response Server** Request-Response Service reports the trace message. **Shutdown** The adapter is shutting down. **Startup** The adapter is starting. **Subscription Service** Subscription Service reports the trace message. **System** This category is not linked to a specific event process. The trace message may be related to a Windows service related messages, memory allocation, file system error, and so on. **TibRvComm** The adapter is communicating with TIBCO Rendezvous. **XML** The adapter is parsing XML documents.
Status Code	Unique code for the message and description. Status codes are identified by a unique number and description. If a trace message includes an error or warn role, the status code documentation includes a resolution. See Status Messages on page 118 for details.
Tracking Identifier	A unique identifier that is stamped on each message by the originating adapter. The tracking identifier remains in effect from a message's beginning to its completion as it is exchanged by TIBCO applications. If the adapter is the termination point of the message, the tracking identifier is not displayed in the trace message. You cannot modify the tracking identifier format or configure what information is displayed.

Status Messages

In environments where multiple applications are used simultaneously, the possible status of messages increases as well. This appendix lists the various messages in numerical order.

Table 46 Status Messages

Status Code	Message	Role	Category	Resolution
TAS400-0001	Incorrect value is specified for server, security library or environment parameter.	errorRole	Connection	Specify correct values for server, security or environment parameter.
TAS400-0002	Error(s) occurred when setting library list for connection. Error(s): *error_details*	errorRole	Connection	Check error(s) listed in the error message and take appropriate action.
TAS400-0003	An exception occurred when trying to connect to server: *error_details*	errorRole	Connection	Check error(s) listed in the error message and take appropriate action.
TAS400-0008	An exception occurred when trying to run the initial program.	errorRole	Connection	Check the initial program.
TAS400-0101	Service *service_name* program call for program *program_name* in library *library_name* ended in error. Error(s): *error_details*	errorRole	PgmCall	Check error(s) listed in the error message and take appropriate action.
TAS400-0102	Service *service_name* encountered an exception while dispatching program call error for program *program_name* in library *library_name*. Error: *error_details*	errorRole	PgmCall	Check error(s) listed in the error message and take appropriate action.
TAS400-0151	Service *service_name* received program call request for program *program_name* in library *library_name*.	infoRole	PgmCall	Indicates normal adapter operation. No action necessary.

Table 46 Status Messages (Cont'd)

Status Code	Message	Role	Category	Resolution
TAS400-0152	Input program parameters read successfully in service *service_name* for program *program_name* in library *library_name*.	infoRole	PgmCall	Indicates normal adapter operation. No action necessary.
TAS400-0153	Output program parameters written successfully in service *service_name* for program *program_name* in library *library_name*.	infoRole	PgmCall	Indicates normal adapter operation. No action necessary.
TAS400-0154	Service *service_name* successfully completed program call for program *program_name* in library *library_name*.	infoRole	PgmCall	Indicates normal adapter operation. No action necessary.
TAS400-0191	Error occurred when calling program.	debugRole	PgmCall	
TAS400-0201	Error occurred when listening to message queue. Error: *error_details*.	errorRole	MonMsg	Check error(s) listed in the error message and take appropriate action.
TAS400-0202	Error occurred when dispatching message. Error: *error_details*.	errorRole	MonMsg	Check error(s) listed in the error message and take appropriate action.
TAS400-0203	Program context for program *program_name* in library *library_name* is already in use. Context details: JobName=*job_name*, JobUser=*job_user*, JobNumber=*job_number*	errorRole	MonMsg	Make sure that dispatcher for monitoring message queue for program call is running in single thread mode.
TAS400-0204	Monitor message service *service_name* for program *program_name* in library *library_name* not found.	errorRole	MonMsg	Specify a valid request-response invocation service name of type 'Monitor Message' in program call service.

Table 46 Status Messages (Cont'd)

Status Code	Message	Role	Category	Resolution
TAS400-0252	Received message from message queue *message_queue_name* in library *library_name*. Message details: JobName=*job_name*, JobUser=*job_user*, JobNumber=*job_number*, MsgId=*message_id*, MsgText=*message_text*	infoRole	MonMsg	Indicates normal adapter operation. No action necessary.
TAS400-0253	Received monitorable message for program *program_name* in library *library_name* with no associated monitor message service. Ignoring message.	infoRole	MonMsg	Indicates normal adapter operation. No action necessary.
TAS400-0254	Request-response invocation service *service_name* received message.	infoRole	MonMsg	Indicates normal adapter operation. No action necessary.
TAS400-0255	Request-response invocation service *service_name* processed message successfully.	infoRole	MonMsg	Indicates normal adapter operation. No action necessary.
TAS400-0291	Error occurred when dispatching message.	debugRole	MonMsg	
TAS400-0292	Error occurred when listening to message queue.	debugRole	MonMsg	
TAS400-0301	Outbound message service *service_name* for message from data queue *data_queue_name* in library *library_name* not found.	errorRole	OutMsg	Outbound message dispatcher received a message for which a request-response invocation service does not exist in this adapter instance. Please check the IBM AS/400 application that sent this message to recover from this error.

Table 46 Status Messages (Cont'd)

Status Code	Message	Role	Category	Resolution
TAS400-0302	Error during Outbound Message operation.	errorRole	OutMsg	Check error(s) listed in the error message and take appropriate action.
TAS400-0303	Request-response invocation service *service_name* received message of unhandled data type *data_type*. Message not processed.	errorRole	OutMsg	The data type specified in the error message is not handled by the adapter. Please change the data type of the variable in error and try request again.
TAS400-0304	Request-response invocation service *service_name* did not find reply data queue *data_queue_name* in library *library_name*.	errorRole	OutMsg	Specify a valid data queue name as reply data queue in request-response invocation service.
TAS400-0351	Received message from data queue *data_queue_name* in library *library_name*. Message details: Version=*adapter_version*, Service=*service_name*, JobName=*job_name*, JobUser=*job_user*, JobNumber=*job_number*, Time=*time_stamp*	infoRole	OutMsg	Indicates normal adapter operation. No action necessary.
TAS400-0352	Request-response invocation service *service_name* received message.	infoRole	OutMsg	Indicates normal adapter operation. No action necessary.
TAS400-0353	Request-response invocation service *service_name* processed message successfully.	infoRole	OutMsg	Indicates normal adapter operation. No action necessary.
TAS400-0391	Error during request-response invocation operation.	debugRole	OutMsg	

Table 46 Status Messages (Cont'd)

Status Code	Message	Role	Category	Resolution
TAS400-0401	Publisher *service_name* received message of unhandled data type *data_type*. Message not processed.	errorRole	Publish	The data type specified in the error message is not handled by the adapter. Please change the data type of the variable in error and try request again.
TAS400-0451	Publisher *service_name* received message.	infoRole	Publish	Indicates normal adapter operation. No action necessary.
TAS400-0452	Output values read successfully.	infoRole	Publish	Indicates normal adapter operation. No action necessary.
TAS400-0453	Publisher *service_name* published message successfully.	infoRole	Publish	Indicates normal adapter operation. No action necessary.
TAS400-0491	Error when publishing message.	debugRole	Publish	
TAS400-0901	Could not map AS/400 data type of '*data_type*'.	errorRole	Marshall	The data type specified in the error message is not handled by the adapter. Please change the data type of the variable in error and try request again.
TAS400-0902	Error occurred when setting sequence value for AS/400 parameter *parameter_name* at index *index_number*. Error: *error_details*	errorRole	Marshall	Check error(s) listed in the error message and take appropriate action.
TAS400-0903	Error occurred when setting data structure value for AS/400 parameter *parameter_name* at element *element_name*. Error: *error_details*	errorRole	Marshall	Check error(s) listed in the error message and take appropriate action.

Table 46 Status Messages (Cont'd)

Status Code	Message	Role	Category	Resolution
TAS400-0904	Error occurred when setting sequence value from AS/400 parameter *parameter_name* at index *index_number*. Error: *error_details*	errorRole	Marshall	Check error(s) listed in the error message and take appropriate action.
TAS400-0905	Error occurred when setting data structure value from AS/400 parameter *parameter_name* at element *element_name*. Error: *error_details*	errorRole	Marshall	Check error(s) listed in the error message and take appropriate action.
TAS400-0907	Variable *variable_name* was set to default value due to error *error_details*.	infoRole	Marshall	Indicates normal adapter operation. No action necessary.
TAS400-9003	Default library for object *object_name* of type *object_type* member *member_name* not found! Error: *error_details*	errorRole	MetaData	Check that the libraries in library list for the adapter instance contains the object mentioned in the error message.
TAS400-9051	Error occurred when deleting temporary list table *table_name* in library *library_name*. Error: *error_details*	infoRole	MetaData	Indicates normal adapter operation. No action necessary.
TAS400-0501	Service *service_name* of type data queue write for data queue *data_queue_name* in library *library_name* ended in error. Error(s): *error_details*	errorRole	DQWrite	Check error(s) listed in the error message and take appropriate action.
TAS400-0502	Service *service_name* did not receive reply in data queue *data_queue_name* in library *library_name* within *timeout* seconds	errorRole	DQWrite	Make sure that data queue request processing jobs is running on IBM System i machine.
TAS400-0551	Service *service_name* received data queue write request for data queue *data_queue_name* in library *library_name*.	infoRole	DQWrite	

Table 46 Status Messages (Cont'd)

Status Code	Message	Role	Category	Resolution
TAS400-0552	Request message written successfully in service *service_name* to data queue *data_queue_name* in library *library_name*.	infoRole	DQWrite	
TAS400-0553	Reply message read successfully in service *service_name* from data queue *data_queue_name* in library *library_name*.	infoRole	DQWrite	
TAS400-0554	Service *service_name* successfully completed data queue write request.	infoRole	DQWrite	
TAS400-0591	Error occurred when writing to data queue.	debugRole	DQWrite	
TAS400-0601	Service *service_name* for spooled files list ended in error. Error(s): *error_details*	errorRole	SpooledFileList	Check error(s) listed in the error message and take appropriate action.
TAS400-0602	Date-time range filter value is incorrect. Start DateTime: *start_date-time*, End DateTime: *end_date-time*	errorRole	SpooledFileList	The date/time range specified for spooled files list operation is incorrect. Please change the date/time range and try request again.
TAS400-0603	Status filter value is incorrect. Status: *status*	errorRole	SpooledFileList	The status value specified for spooled files list operation is incorrect. Please change the status value and try request again.
TAS400-0651	Service *service_name* received spooled files list request.	infoRole	SpooledFileList	
TAS400-0652	Spooled files list filters read and set successfully in service *service_name*.	infoRole	SpooledFileList	

Table 46 Status Messages (Cont'd)

Status Code	Message	Role	Category	Resolution
TAS400-0653	Service *service_name* successfully completed spooled files list request.	infoRole	SpooledFileList	
TAS400-0691	Error occurred in spooled files list operation	debugRole	SpooledFileList	
TAS400-0701	Service *service_name* for spooled file to PDF ended in error. Error(s): *error_details*	errorRole	SpooledFilePDF	Check error(s) listed in the error message and take appropriate action.
TAS400-0702	Required parameter *parameter_name* has incorrect value.	errorRole	SpooledFilePDF	The value of parameter name specified in error has incorrect value. Please change the value for parameter and try request again.
TAS400-0703	Parameter *parameter_name* has negative value.	errorRole	SpooledFilePDF	The value of parameter name specified in error has negative value. Please change the value for parameter and try request again.
TAS400-0704	Spooled file printer device type is not *SCS.	errorRole	SpooledFilePDF	The spooled file is not a *SCS spooled file. Please specify a *SCS type spooled file and try request again.
TAS400-0705	Spooled file name '*spooled_file_name*', number *spooled_file_number* for job *job_name/job_user/job_number* not found.	errorRole	SpooledFilePDF	The spooled file specified for this operation was not found. Please specify an existing spooled file and try request again.
TAS400-0751	Service *service_name* received spooled file to PDF request.	infoRole	SpooledFilePDF	
TAS400-0752	Service *service_name* successfully completed spooled file to PDF request.	infoRole	SpooledFilePDF	

Table 46 Status Messages (Cont'd)

Status Code	Message	Role	Category	Resolution
TAS400-0791	Error occurred in spooled file to PDF operation	debugRole	SpooledFilePDF	

Index

A

ActiveMatrix SOA project 53
Adapter Name 91, 92, 97, 98, 100, 101
Adapter Projects
  create 6
  import 9
agents 76
alerts 76
AMX_HOME xiv
Attribute Name 93
Attribute Value 93
auto-discovery process, TIBCO Hawk 78

B

business object 28

C

Certified Name 97
Class Microagent Name field, adapter 36
CM Timeout 97
Command 92
command line arguments 92
Component Name 91
Component Type 91
Composite Editor 56
Configuration URL 92
connection 26
converting
  Business Objects into AESchemas 33
creating an ActiveMatrix SOA project 54
creating service assembly 60
customer support xvii

D

Daemon 97
Data Queue for Publish 40
Deployment 72
Description 35, 40, 41, 43, 47, 91
DirLedger 72
DirTrace 72
Domain 72
DQWrite Operation 32

E

Element Name 93
Enable Class Microagent 36
Enable Standard Microagent 36

F

Failure 87, 88, 89
File Size 109

G

Generate Operation BO 30
generating WSDL file 55
GetSubTotalBy 87
global variables 6

H

Host 98

I

IBM i Connection 26
import
  business object 28
  projects 9
  TIBCO Designer project 7
inbound thread count 35
Initial Program To Run 39
Instance ID 91, 92, 97, 98, 100, 101
Instance Name 35

J

JmsProviderUrl 73
JmsSslProviderUrl 73, 73, 73, 73

L

Last Confirmed 108
Last Sent Message 107
Ledger File 97
Line 93, 108
LineIndex 88, 89
Listener Session Name 103, 107, 111

M

MaxQueueSize 96
MeasurementInterva 88
MeasurementInterval 87, 89, 96, 99
message queue 47
Messages Received 98
Messages Sent 98
microagent methods supported 82
Microagent Session field, adapter 36
Microagents 79
multithreading pool 35

N

Name 39, 41, 43, 47, 87, 95
Network 97
New Errors 98

O

Object Group 30
Object Parameter Group 30
Operation 88, 88, 89
Operation BO 32
OutBound Operation 33

P

Peek/Read 40
Predefined variables 70
Process ID 98
ProgramCall Operation 32
Property 93
Publisher Name 103, 111

Q

QueueCount 96
QueueID 96
QueueType 96

R

RemoteRvDaemon 73
Repository Connection 92
Role Name 100, 109
Roles 100
RPC Client 47
RvaHost 73
RvaPort 73
RvDaemon 73
RvNetwork 73
RvService 73

S

Schema Name 89
Service 97
Service Assembly
  deployment
    configure 64
    deploy 65
    start 66
    stop 66
    upload 64
Service Name 88, 89, 89
Session Name 91, 97, 97, 103, 107, 107, 111
Session Type 97
setTraceSink() 85
Sink Name 100, 100, 109
Sink Type 100
Source File Group 30
Standard Microagent Name field, adapter 36
Subject 107, 107
Subscriber 39
substitution variables 6
Success 87, 88, 89
support, contacting xvii
Synchronous? 97

T

TaskCount 99
TaskType 99
technical support xvii
ThreadID 99
ThreadType 99
TIBCO Hawk
  background information 76
  enterprise monitor components 76
TIBCO_HOME xiv
Total 87, 88, 89
Total Errors 98
Total Messages 107
Total Size 107
Tracing 114, 114
Tracing Levels and Fields 116, 116

U

UnacknowledgedMessages 108
Uptime 98

V

Value 95
Version 101

W

WSDL 55

X

XML Schema File Group 30