TIBCO ActiveMatrix® Adapter for Database (TIBCO Business Studio™)

User's Guide

Software Release 1.1
May 2014

Two-Second Advantage®

Important Information

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

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

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

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

Contents

Figures

Tables

Preface

TIBCO ActiveMatrix Adapter Framework provides a visual, model-driven development environment for configuring supported TIBCO adapters in TIBCO Business Studio and integrating them with TIBCO ActiveMatrix BusinessWorks applications and processes. The product also provides administration capabilities for deploying the TIBCO adapters developed in TIBCO Business Studio to runtime environments.

This document explains how to configure, deploy, manage, and monitor TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) projects by using TIBCO ActiveMatrix Adapter Framework in TIBCO ActiveMatrix BusinessWorks.

Topics

Related Documentation

This section lists documentation resources you may find useful.

TIBCO ActiveMatrix Adapter Framework Documentation

The following documents form the TIBCO ActiveMatrix Adapter Framework documentation set:

- *TIBCO ActiveMatrix Adapter Administration* Read this manual for instructions on how to deploy and manage adapter projects.
- *TIBCO ActiveMatrix Adapter Reference* Read this manual for reference information of TIBCO ActiveMatrix Adapter activities.
- *TIBCO ActiveMatrix Adapter Framework Installation* Read this manual for instructions on site preparation and installation.
- *TIBCO ActiveMatrix Adapter Framework Release Notes* Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.

TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) Documentation

The following documents form the TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) documentation set:

- *TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) User's Guide* Read this manual for instructions on how to create, configure, and deploy adapter projects.
- *TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) Examples* Read this manual to work through the examples provided with the adapter.

Other TIBCO Product Documentation

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

Table 1 TIBCO Products

TIBCO Product	Description
TIBCO ActiveMatrix® Adapter for Database	This product is a bidirectional gateway between databases and applications configured for the TIBCO environment. The adapter supports publication, subscription, and request-response interactions. It is the runtime component for TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio).
TIBCO ActiveMatrix BusinessWorks™	This product provides an integration platform that enables companies to rapidly integrate systems and automate business processes.
TIBCO® Adapter SDK	This product provides a class library that helps developers to implement an adapter with minimal effort.
TIBCO Business Studio™	This product provides capabilities that help business analysts document business processes, solution engineers implement business processes, and SOA developers create composite applications.
TIBCO® Database Drivers Supplement	This product provides the licensed DataDirect database drivers that can be used with certain TIBCO products.
TIBCO Designer™	This product provides a graphical user interface to create TIBCO ActiveMatrix BusinessWorks process definitions, or create or modify TIBCO Adapter configurations.
TIBCO® Enterprise Administrator	This product provides capabilities for managing users, monitoring machines, and deploying and managing applications created in TIBCO Business Studio.
TIBCO Enterprise Message Service™	This product sends messages from your applications in a format that conforms to the Java Messaging Service (JMS) specification.
TIBCO Hawk®	This product is for monitoring and managing distributed applications and systems throughout the enterprise.
TIBCO Rendezvous®	This product uses messages to enable distributed application programs to communicate across a wide variety of hardware platforms and programming languages.
TIBCO Runtime Agent™	This product is a bundle of TIBCO software and third-party software that is required to run many TIBCO applications such as TIBCO ActiveMatrix BusinessWorks, TIBCO Adapters, and so on.

Typographical Conventions

The following typographical conventions are used in this manual.

Table 2 General Typographical Conventions

Convention	Use
ENV_NAME *TIBCO_HOME*	TIBCO products are installed into an installation environment. A product installed into an installation environment does not access components in other installation environments. Incompatible products and multiple instances of the same product must be installed into different installation environments. An installation environment consists of the following properties: • **Name** Identifies the installation environment. This name is referenced in documentation as *ENV_NAME*. The default value is `TIBCO_HOME`. • **Directory** The folder into which the product is installed. This folder is referenced in documentation as *TIBCO_HOME*.
CONFIG_HOME	A TIBCO configuration home stores configuration data generated by TIBCO products. Configuration data can include sample scripts, session data, configured binaries, logs, and so on. This folder is referenced in documentation as *CONFIG_HOME*. The default location of the folder is *USER_HOME*`/`*ENV_NAME*`/data`. For example, on Microsoft Windows, the default location is `C:\Documents and Settings\`*UserName*`\Application Data\`*ENV_NAME*`\data`.
TIB_ADADB_HOME	TIBCO ActiveMatrix Adapter for Database is installed into a directory within a *TIBCO_HOME* directory. This directory is referenced in documentation as *TIB_ADADB_HOME*. The default value of *TIB_ADADB_HOME* depends on the operating system. For example, on Windows systems, the default value is `C:\tibco\adapter\adadb\`*ReleaseNumber*.
`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.

Table 2 General Typographical Conventions (Cont'd)

Convention	Use
`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]`
italic font	Italic font is used in the following ways: • To indicate a document title. For example: See *TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) User's Guide*. • 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 property 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 3 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 param1 | param2 | param3`

`{ }`	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}`

Connecting with TIBCO Resources

How to Join TIBCOmmunity

TIBCOmmunity is an online destination for TIBCO customers, partners, and resident experts. It is a place to share and access the collective experience of the TIBCO community. TIBCOmmunity offers forums, blogs, and access to a variety of resources. To register, go to http://www.tibcommunity.com.

How to Access TIBCO Documentation

You can access TIBCO documentation here:

http://docs.tibco.com

How to Contact TIBCO Support

For comments or problems with this manual or the software it addresses, 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 to Adapter Life Cycle

The adapter life cycle includes preparation, design and configuration, deployment to the production environment, and management and monitoring. TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) provides you with maximum configurability and an improved user experience during each phase of the life cycle.

In this document, TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) is also referred to as the adapter user interface in TIBCO Business Studio.

Topics

Overview of Using TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio)

Using the adapter user interface in TIBCO Business Studio in a whole adapter life cycle involves the following tasks:

- Preparation, page 2
- Design and Configuration, page 2
- Deployment, page 4
- Production, page 4

For basic steps on how to configure and run the adapter, see Chapter 2, Getting Started, on page 9.

Preparation

Before design and configuration, install TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio), and all required software products, and ensure each product is operating correctly. See the installation guide for each required product for details.

You also need to prepare TIBCO Business Studio for use with the adapter. See Chapter 2, Getting Started, on page 9. This chapter also includes a sample tutorial on how to set up and test the adapter.

Design and Configuration

The adapter user interface in TIBCO Business Studio provides you with a variety of flexible ways to design and configure adapters and processes.

This guide introduces the configuration workflows based on the following entry points:

- Adapter Configurations, page 3
- Database Tables and Stored Procedures, page 3
- BusinessWorks Processes, page 3
- Migration from TIBCO Designer, page 3

Adapter Configurations

Workflows started from adapter configurations are for users with TIBCO Designer background.

See Chapter 3, Working with Adapter Configurations and Services, on page 31 for details.

In addition to the general service configuration, each of the three services the adapter supports has its specific options to set up, see the following chapters for details:

- Chapter 7, Configuring Publication Services, on page 101
- Chapter 8, Configuring Subscription Services, page 141
- Chapter 9, Configuring Request-Response Services, page 159

The adapter supports a variety of performance tuning approaches. See Chapter 10, Performance Tuning, on page 185.

Database Tables and Stored Procedures

Workflows started from database tables or stored procedures are for users who are familiar with data in the database. With the adapter user interface, you can fetch tables, stored procedures, or stored functions from a database for use in different adapter configurations.

See Chapter 4, Working with Tables and Stored Procedures, on page 75 for details.

BusinessWorks Processes

Business processes are used for capturing and managing the flow of business information in an enterprise between different data sources and destinations.

See Chapter 5, Working with BusinessWorks Processes, on page 85 for details.

Migration from TIBCO Designer

The Migration Tool in TIBCO Business Studio can help you migrate a project designed in TIBCO Designer with TIBCO ActiveMatrix BusinessWorks 5.x to TIBCO ActiveMatrix BusinessWorks 6.

See Chapter 6, Migrating an Adapter Project Created in TIBCO Designer, on page 93 for details.

Deployment

Once the adapter project is ready for production, you need to first generate an Enterprise Archive (EAR) file for the project from TIBCO Business Studio. Then, use the TIBCO ActiveMatrix Adapter Framework utilities to deploy and manage adapters, and use TIBCO ActiveMatrix BusinessWorks utilities to deploy and manage the processes of the application at the same time.

See *TIBCO ActiveMatrix Adapter Administration* for details.

Production

After you deploy your project, you can use TIBCO Enterprise Administrator to start, stop, and perform more management functions with the deployed application.

See *TIBCO ActiveMatrix Adapter Administration* for details.

Advanced Topics

For other configuration and deployment topics, see Chapter 11, Advanced Topics, on page 203.

Features of TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio)

The adapter allows data changes in a database to be sent as they occur to other databases and applications. It extends publish-subscribe and request-response technology to databases, making multiple levels of delivery services available to applications that need access to these databases. JDBC-compliant databases, such as Oracle, DB2, Sybase, Microsoft SQL Server, MySQL, PostgreSQL, and Teradata are supported. Even though the adapter cannot run on z/OS or iSeries systems, it can remotely connect to a DB2 database running on these systems.

The adapter supports the following features:

- Automatically publish data when rows in pre-specified database tables are inserted, updated, or deleted:
  - Monitor database changes by using periodic polling or notification by an alerting mechanism. See Specifying the Polling Method on page 123.
  - Publish data by creating a copy (by value). See Publish by Value on page 129.
  - Publish by reference (publish data directly from the source database table without first copying the data from the source table to a publishing table). See Publish by Reference on page 130.
  - Publish on parameterized subjects or destinations, which allows a subject or destination to be created based on the contents of one or more table columns. See Subject and Destination Names on page 205

The subject names mentioned in this document are referred to as subject names when using the TIBCO Rendezvous transport type, but are referred to as destination names when using the JMS (Java Messaging Service) transport type.

  - Update both parent and child tables within a publication. Both the parent row and all related child rows will be published if the user has set up to publish child data. See Adding Child Tables on page 109.
  - Specify that a group of rows fetched from the publishing table is sent in a single message.
  - Preregister certified subscribers with a certified publisher. See Preregistering a Certified Subscriber on page 209.

- Automatically subscribe to data and insert, update or delete the data in pre-specified tables in a database:
  - By default, you can subscribe a group of rows set in a single database transaction.
  - Collect the changed data in both parent and child tables within a subscription. Both the parent row and all related child rows will be updated if the user has set up to subscribe child data. See Subscription Options Tab on page 143.
  - Accumulate received messages and perform bulk inserts based on pre-configured "Bulk Insert Size" value in database transactions to reduce database traffic. See Batch Processing in Subscription Services on page 199.
  - Perform batch commits based on pre-configured "Batch Commit Size" and "Batch Commit Timeout" values. See Batch Processing in Subscription Services on page 199.
  - Subscribe data by using wildcard subject names. See Preregistering a Certified Subscriber on page 209.
- Use Request-Response Service to subscribe SQL statements, execute stored procedures, or both on a specified subject:
  - Execute stored procedures or functions with `IN` parameters, `OUT` parameters, or both. (Binary `OUT` parameters are not supported.) See Using the REF Data Type with Oracle Databases on page 182.
  - Subscribe to messages in Request-Response mode, and specify reply subject to receive returned messages after the database execution. See Using Request-Reply Mode on page 164.
  - Support RPC (Remote Procedure Calls). The adapter can act as an operation server providing a simple means for a client to execute a single or batch of SQL statements. See Using RPC Mode on page 173.
- Change published messages for a Publication or Subscription Service by customizing the supplied callout library. See Using the User Callout Java Library on page 213.

- Rely on standards:
  - — Connect to multiple database vendors using JDBC drivers. See Configuring the Database Connection on page 37.
  - — Interact with other TIBCO products through the use of TIBCO Adapter SDK software.
  - — Monitor your adapter service using TIBCO Hawk software. See Configuring Monitoring Options on page 65.
  - — Choose message transports: TIBCO Rendezvous or JMS (Java Messaging Service). See Configuring Transports on page 46.

Chapter 2 Getting Started

Prior to using TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) for development, you need to do some preparation in TIBCO Business Studio, including creating a project and configuring the preferences and module properties for the adapter. This chapter includes a sample tutorial to help you familiar yourself with the development and test procedures of the adapter using TIBCO Business Studio.

Topics

Preparing the Development Environment

You need to perform these tasks before configuring and using the adapter:

- Creating a Project, page 10
- Configuring Adapter Preferences, page 11
- Configuring Module Properties, page 12

When you finish the preparation, you can start to design and configure adapters and processes.

Creating a Project

BusinessWorks Application Modules are Eclipse projects that are configured for BusinessWorks. A project contains folders and files and is used for version management, sharing, and resource management. You must create an application module for adapter configurations.

To create a BusinessWorks Application Module:

1. Open the **New BusinessWorks Application Module** dialog using one of the following ways:

   — Select **File > New > BusinessWorks Resources** from the TIBCO Business Studio main menu. In the **BusinessWorks Resource** wizard, select **BusinessWorks Application Module** and click **Next**.

   — Click the **Create a New BusinessWorks Application Module** button from the toolbar.

2. Specify the project name. Do not use spaces or special characters in the name.
3. To use the default location, leave the check boxes selected.
4. Click **Finish**.

Two project folders are created with the given project name. One folder is for the BusinessWorks Application Module, and another, suffixed with *application_module_name*.application, is for the BusinessWorks Application that packages the module.

The Project Explorer view displays all projects with their folders and files and lets you manage these resources.

See the TIBCO ActiveMatrix BusinessWorks documentation for more information about projects.

Configuring Adapter Preferences

Use the **Window > Preferences** dialog to set the preferred settings for the adapter configuration.

From the left panel of the dialog, click **TIBCO Adapter > Adapter for Database** to open the preferences for the adapter.

You can change the following preferences:

Table 4 Adapter Preferences

Setting	Description
Adapter Launcher (General tab)	
Working Directory	The Adapter Launcher creates the necessary runtime and support files required by the adapter in this directory. **See also**: Working Directory on page 73
Default Service Type (General tab)	
Default Service Type	Default type of the service created.
JDBC Driver Path (General tab)	
JDBC Libraries Path	Folder which contains JDBC drivers. If left blank, TIBCO Database Drivers installed under *TIBCO_HOME* will be used.
Default Transport Settings (Transport tab)	
Default Transport	Default transport type used. **See also**: Supported Sessions on page 46.
Rendezvous	The preferences include: • Quality of Service for Inbound • Quality of Service for Outbound

Table 4 Adapter Preferences (Cont'd)

Setting	Description
JMS	The preferences include: • Connection Factory **See also**: Connection Factory on page 52 • Delivery Mode for Inbound • Delivery Mode for Outbound **See also**: Guideline for Configuring the Delivery Mode (JMS Only) on page 45.

Configuring Module Properties

Module Properties provide an easy way to set default values for use throughout your project. The variable substitution mechanism can override module properties, predefined in the project.

A number of module properties are predefined in TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio). See Appendix C, Predefined Module Properties, on page 255.

Module properties used in the adapter are called Global Variables when using TIBCO Designer with TIBCO ActiveMatrix BusinessWorks 5.x.

Specify Module Properties

You can specify module properties using one of the following ways:

- Use the Module Properties Editor, page 12
- Use the Binding Editor, page 13

Use the Module Properties Editor

You can add, specify, and group module properties by using the Module Properties editor in TIBCO Business Studio.

1. In the Project Explorer view, select **Module Descriptors > Module Properties** from the project.
2. Click **New Property** to create a new module property.
3. (Optional) Click **New Group** to create a group for a set of module properties.

Use the Binding Editor

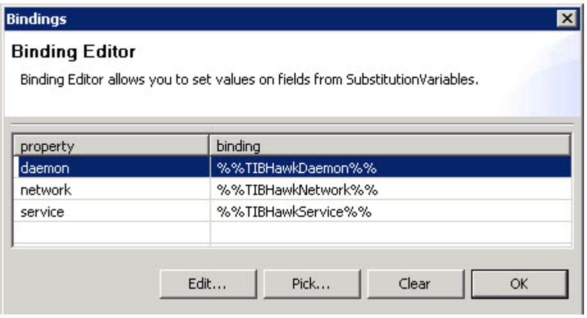
You can also use Binding Editor to specify module properties. Binding Editor is a tool to edit, pick, and clear module properties of every adapter configuration.

You can select multiple module properties and edit each property using the Binding Editor. The global variable type has to match the selected module property type.

You cannot edit, pick or clear module property if the associated input field is grey.

Click the **Binding Editor** icon to open the Bindings dialog.

The following figure shows a Bindings dialog example:



Bindings

Binding Editor

Binding Editor allows you to set values on fields from SubstitutionVariables.

property	binding
daemon	%%TIBHawkDaemon%%
network	%%TIBHawkNetwork%%
service	%%TIBHawkService%%

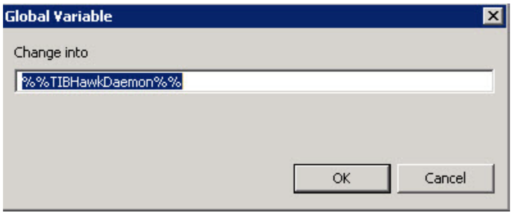
Binding Editor provides the following three functions to configure module properties:

- Edit Module Properties

  a. Select one of the properties in the Binding Editor dialog and click **Edit**.

  b. In the Global Variable dialog, enter a new global variable in the format of %%*Property_Name*%%.

     The following figure shows a Global Variable dialog example:



Global Variable

Change into

%%TIBHawkDaemon%%

  c. Click **OK** to save the edited property.

- Pick Module Properties

a. Select one of the properties in the Binding Editor and click **Pick**.

b. Enter a search expression in to the **Select an item to open** field of the Select String Substitution Variable dialog.

This search expression is used to filter out matching variables.

c. Select the matching variable from the Matching Items panel.

d. Click **OK** to format your variable in the %%*Property_Name*%% format in the Global Variable dialog.

e. Click **OK** to save the picked property.

- Clear Module Properties

a. Select one of the properties in the Binding Editor.

b. Click **Clear** to clear the variable value in the Binding column.

c. Click **OK** to save the property.

Define a Module Property with a Combined Value

You can define a module property with a combined value of other module properties and some literal value. But this type of definition is only supported in an adapter configuration, not in a BusinessWorks process.

For example, you can define the following two module properties in TIBCO Business Studio:

Name: *port* Value: **`8080`**

Name: *URL* Value: **`http://localhost:`***port*

Based on these definitions, the literal value of *URL* is **`http://localhost:8080`**. But this definition is not supported in a BusinessWorks process.

Use Module Properties

To use the module properties in any of the resource fields, enter the property name surrounded by `%%` on both sides. For example, you must enter `%%UserName%%` in the User Name field to use the UserName property.

Precedence of Module Properties

You can also specify variable values in a TRA properties file. A variable value set in the properties file overrides the same variable set as a module property in TIBCO Business Studio.

Getting Started Tutorial

In this exercise, you create tables in your database and configure a publisher adapter and a subscriber adapter. Then you modify a table and observe how the publisher adapter and subscriber adapter handle the changes and update the subscription table.

The exercise uses a project created in TIBCO Designer. The tutorial guides you to migrate the project to TIBCO Business Studio.

Tutorial Overview

Tables and Data Flow

In this tutorial, you need to deal with the following tables for exchanging data:

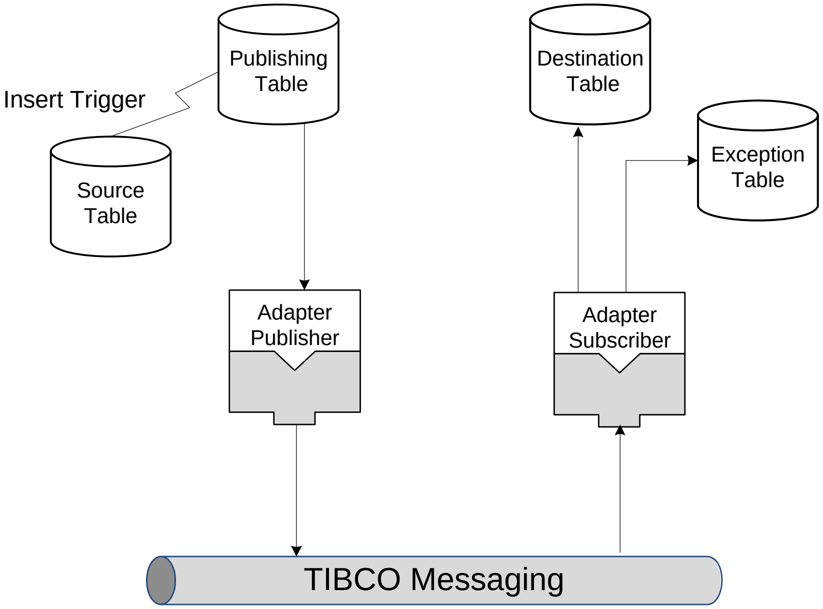
- **Source table** For adding new data.
- **Publishing table** The new data from the source table is copied into the publishing table by using the trigger set on the source table. This table has additional columns (prefixed by `ADB_`) that are used by the adapter.
- **Destination table** The adapter can update the new data to this table.
- **Exception table** The adapter can write any error that occurs during the subscription.
- **Opaque exception table** The adapter can write any error that occurs while writing errors to the exception table.

When you add data to the source table, the following actions occur:

1. For the Publication Service, when data changes in the source table, if you set Publish by Value as storage mode, then the insert, update, delete, and upsert actions fire a trigger and the inserted row is copied to the publishing table. If you set Publish by Reference as storage mode, only the rows inserted in the primary key column and the user defined key column can be copied to the publishing table.
2. The publisher adapter polls the publishing table to check if any new row has been inserted. Newly inserted rows are fetched by using JDBC, packed into a message, and published.
3. The subscriber adapter listens for messages. When a message arrives, the subscriber adapter inserts it into the destination table by using JDBC.

The following figure illustrates the actions.

Figure 1 Diagram of Publish-Subscribe Steps



Location of the Getting Started Example Files

TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) does not include separate shipped examples. The tutorial uses the examples installed with TIBCO ActiveMatrix Adapter for Database 7.0. The preconfigured examples this tutorial uses are located in the *TIB_ADADB_HOME*`\demo\demo1` directory.

Required Platform and Software

You can perform this exercise on any supported operating system, with JMS transport type. The databases you can use with the example include Oracle, Microsoft SQL, and DB2 databases. This guide describes how to run the tutorial on an Oracle database from a Windows platform.

Because the project file is a `.dat` file, you need to install TIBCO Designer to import the file and convert it into the folder structure recognized by TIBCO Business Studio.

Working through the Tutorial

The tutorial includes the following tasks:

- Task A, Create Database Tables, page 17

- Task B, Prepare the Project in TIBCO Designer, page 18
- Task C, Migrate the Project to TIBCO Business Studio, page 20
- Task D, Set the Adapter Preferences, page 21
- Task E, Configure the Adapter Configurations, page 23
- Task F, Test the Adapter Configurations, page 25
- Task G, Clean up the Database, page 29

Task A Create Database Tables

To create database tables, perform the following steps:

1. Open a command window and change directory to the `demo1` subdirectory. For example:

   ```
   > cd TIB_ADADB_HOME\demo\demo1
   ```

2. Execute the `demo1_`*databaseVendor*`.sql` script in the subdirectory to create the tables for your database. Use your environment-specific user ID, password, and database service. For example:

   ```
   > sqlplus userid/password@dbService @demo1_ora.sql
   ```

   The script creates the items and displays the status. For example:

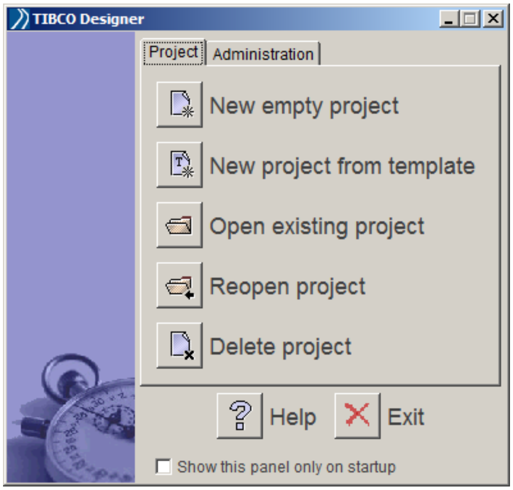
```
TIB_ADADB_HOME\demo\demo1>sqlplus karlh/karlh@ORCL @demo1_ora.sql

SQL*Plus: Release 11.2.0.1.0 Production on Mon Mar 18 17:23:39 2013

Copyright (c) 1982, 2010, Oracle.  All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application
Testing options

Table created.
Table created.
Table created.
Index created.
Index created.
Index created.
Sequence created.
Trigger created.
Table created.
Table created.
Table created.
Table created.
Index created.
Index created.
Sequence created.
```

```
Trigger created.
Table created.
Commit complete.
SQL>
```

Task B Prepare the Project in TIBCO Designer

Import the `ADBDemo1_ora.dat` file under the *TIB_ADADB_HOME*`\demo\demo1` directory into TIBCO Designer:

1. Open TIBCO Designer.
2. In the TIBCO Designer dialog, click the **New Empty Project** button in the Project tab.



3. In the Save Project dialog, specify the following fields for the project in the Multi-File Project tab:

   — **Project Directory**: the location you want to save the project.

   — **TIBCO Message Encoding**: the encoding used in TIBCO message framework. Select **UTF-8** from the list.

   Click **OK** to save the project.

4. From the main menu, click **Project > Import Full Project**.
5. In the Import Project dialog, click the **Local Repository** tab and specify the following fields:

   — **User**: default to the current user.

   — **Project File**: the location of the `.dat` file.

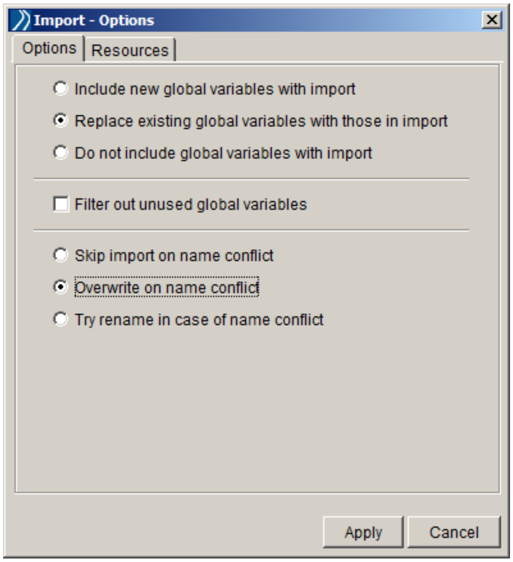
Click **OK** to import the project.

6. In the Import - Options dialog, click the following radio buttons:

   — **Replace existing global variables with those in import**

   — **Overwrite on name conflict**

   Click **Apply**.



Import - Options

Options | Resources

○ Include new global variables with import

● Replace existing global variables with those in import

○ Do not include global variables with import

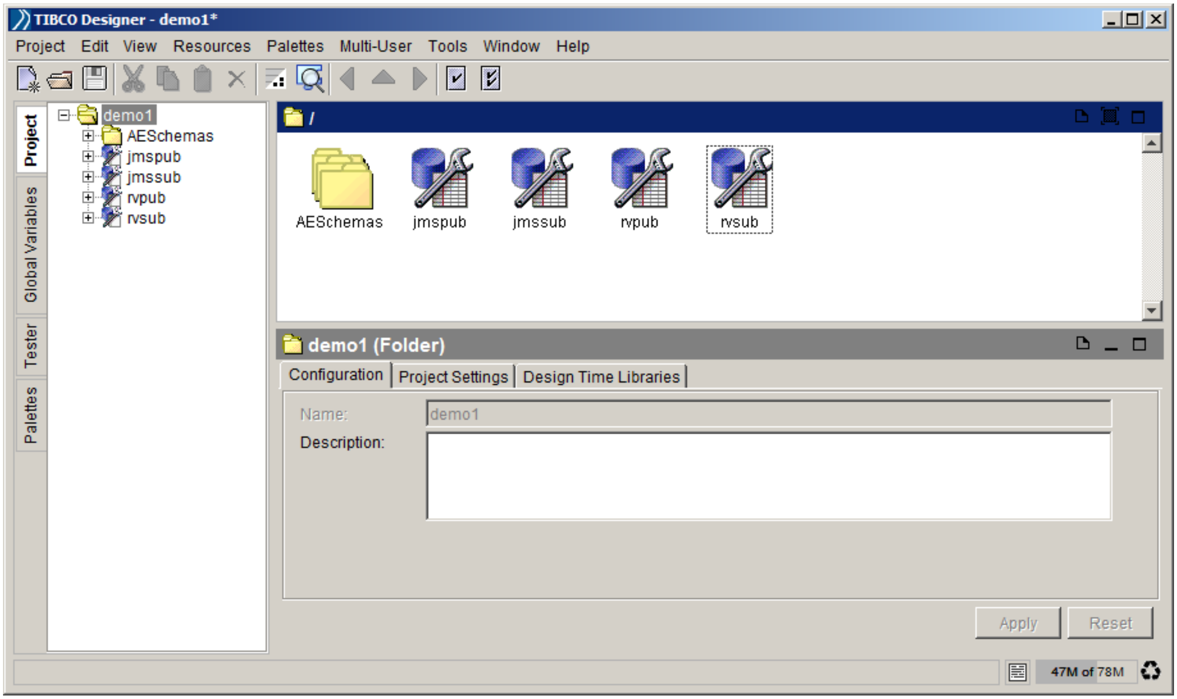
□ Filter out unused global variables

○ Skip import on name conflict

● Overwrite on name conflict

○ Try rename in case of name conflict

7. After the project is imported, click **Save** from the toolbar.



Name: demo1
Description:

8. Exit TIBCO Designer.

Task C Migrate the Project to TIBCO Business Studio

To migrate the project to TIBCO Business Studio:

1. Open TIBCO Business Studio.

2. From the main menu, click **Project > Migrate BW Projects**.

3. In the BusinessWorks Migration Tool dialog:

   a. Specify the following fields:

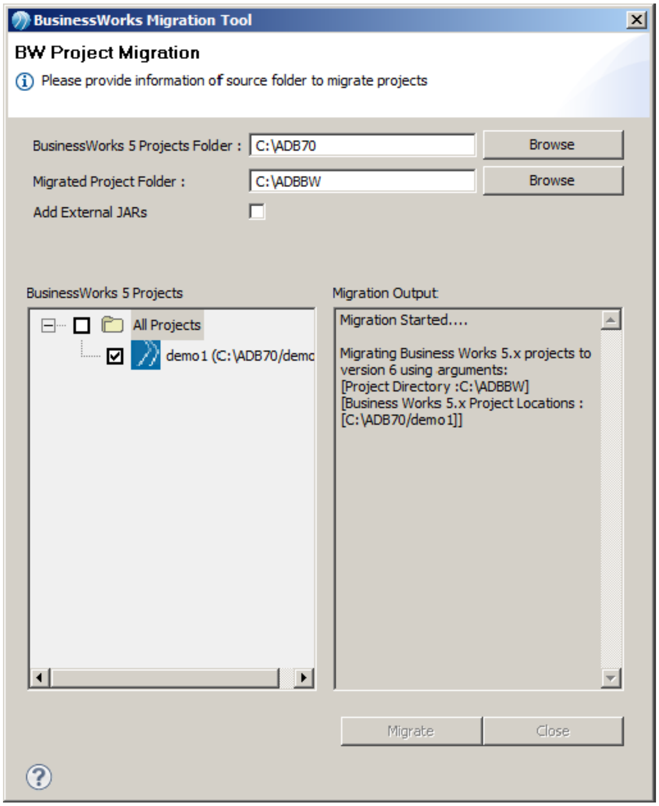
      - **BusinessWorks 5 Projects Folder**: the location you save the project in Task B.

      - **Migrated Project Folder**: the location you want to save the project.

   b. In the BusinessWorks 5 Projects panel, select the check box for the project you want to migrate. In this exercise, select **demo1**.

   c. Click **Migrate**.

   d. When the migration finishes, click **Close**.



BusinessWorks Migration Tool

BW Project Migration

Please provide information of source folder to migrate projects

BusinessWorks 5 Projects Folder : C:\ADB70

Migrated Project Folder : C:\ADBBW

Add External JARs □

BusinessWorks 5 Projects

□ All Projects

☑ demo1 (C:\ADB70/demo

Migration Output

```
Migration Started....

Migrating Business Works 5.x projects to
version 6 using arguments:
[Project Directory :C:\ADBBW]
[Business Works 5.x Project Locations :
[C:\ADB70/demo1]]
```

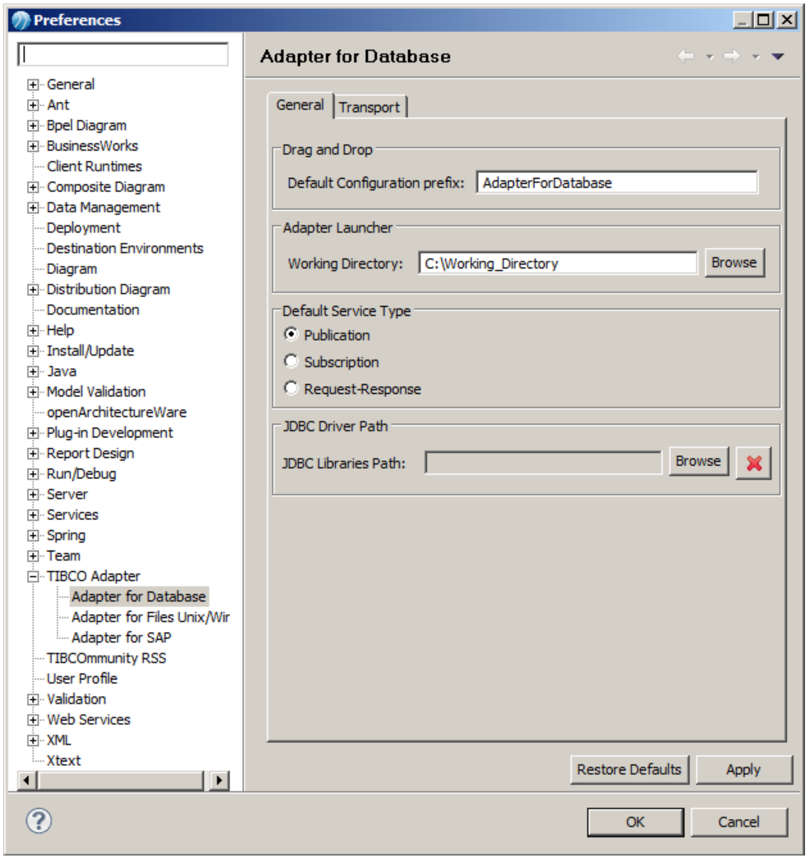
4. Click **Save** in the toolbar of TIBCO Business Studio.

Task D Set the Adapter Preferences

To set the preferences for the adapter:

1. Open TIBCO Business Studio.

2. From the main menu, click **Window > Preferences**.
3. In the Preferences dialog, click **TIBCO Adapter > Adapter for Database** to open the preferences for the adapter.



Drag and Drop

Default Configuration prefix: AdapterForDatabase

Adapter Launcher

Working Directory: C:\Working_Directory

Default Service Type

● Publication

○ Subscription

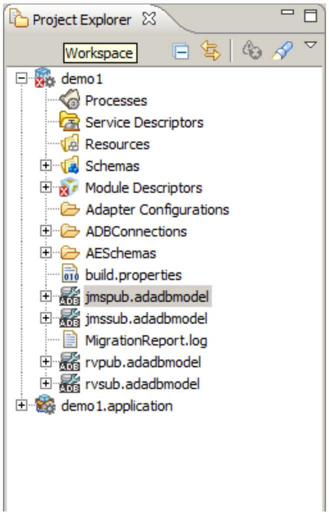
○ Request-Response

JDBC Driver Path

JDBC Libraries Path:

4. Click the **General** tab and specify the **Working Directory** field in the Adapter Launcher panel. Provide the directory where you want to save the runtime adapter.

5. Click **OK**. The migrated project is displayed in the Project Explorer.



Project Explorer
Workspace

- demo1
  - Processes
  - Service Descriptors
  - Resources
  - Schemas
  - Module Descriptors
  - Adapter Configurations
  - ADBConnections
  - AESchemas
  - build.properties
  - jmspub.adadbmodel
  - jmssub.adadbmodel
  - MigrationReport.log
  - rvpub.adadbmodel
  - rvsub.adadbmodel
- demo1.application

Task E Configure the Adapter Configurations

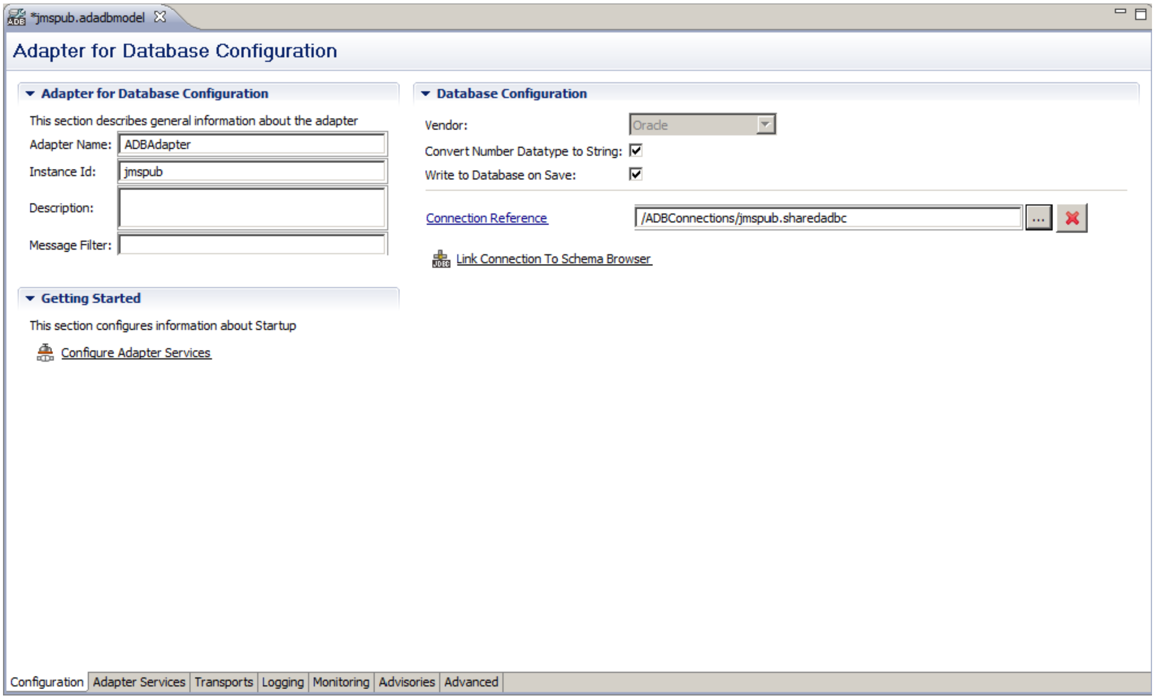
The adapter configurations in the project are preconfigured. But you need to update the connection information for each adapter configuration.

1. From the Project Explorer, expand the migrated project **demo1** and double click **`jmspub.adadbmodel`** to open the adapter configuration editor. This is the file that stores the information for the `jmspub` adapter configuration.

2. In the Adapter Configuration editor, change the connection reference:

   a. Click the **Configuration** tab.

      The following figure shows the **jmspub** adapter configuration.



*jmspub.adadbmodel

Adapter for Database Configuration

Adapter for Database Configuration

This section describes general information about the adapter

Adapter Name: ADBAdapter

Instance Id: jmspub

Description:

Message Filter:

Getting Started

This section configures information about Startup

Configure Adapter Services

Database Configuration

Vendor: Oracle

Convert Number Datatype to String: ☑

Write to Database on Save: ☑

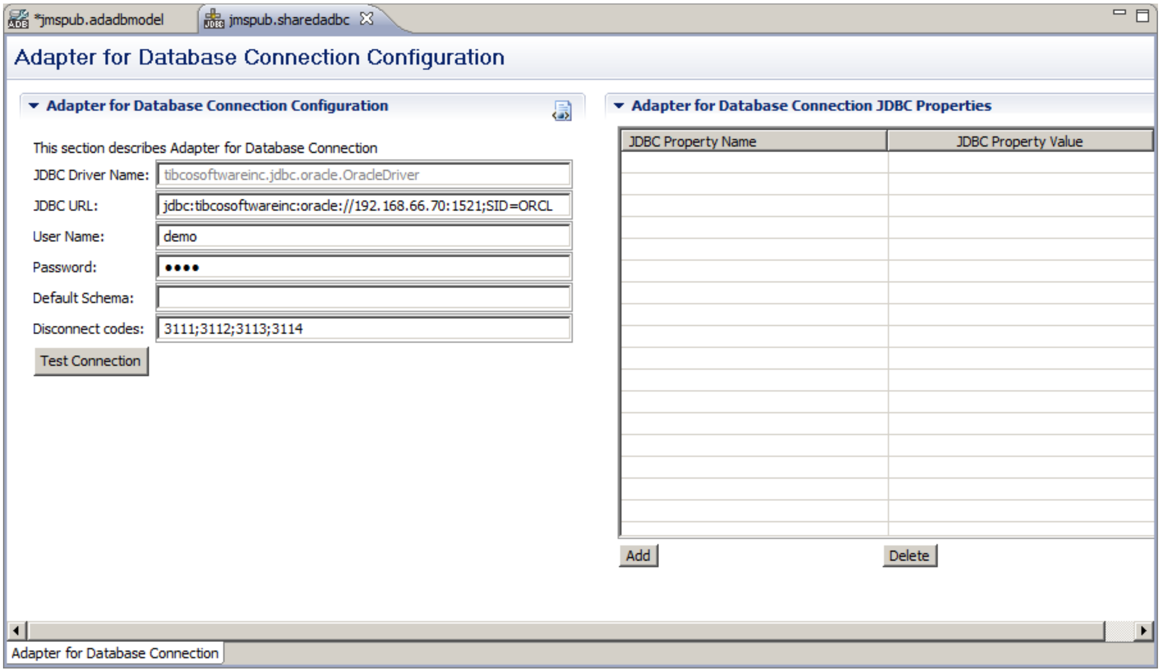
Connection Reference: /ADBConnections/jmspub.sharedadbc

Link Connection To Schema Browser

b. In the Database Configuration panel, click the **Connection Reference** link.

c. In the opened **jmspub.sharedadbc** Adapter for Database Connection editor, update the configuration information.

The following figure shows a sample configuration:


*jmspub.adadbmodel
jmspub.sharedadbc
Adapter for Database Connection JDBC Properties
Add
Delete
Test Connection
Adapter for Database Connection

Adapter for Database Connection Configuration

Adapter for Database Connection Configuration

This section describes Adapter for Database Connection

JDBC Driver Name: tibcosoftwareinc.jdbc.oracle.OracleDriver

JDBC URL: jdbc:tibcosoftwareinc:oracle://192.168.66.70:1521;SID=ORCL

User Name: demo

Password: ••••

Default Schema:

Disconnect codes: 3111;3112;3113;3114

JDBC Property Name	JDBC Property Value

3. Click **Save** in the toolbar of TIBCO Business Studio.

Repeat the procedure for the **`jmssub.adadbmodel`** adapter configuration.

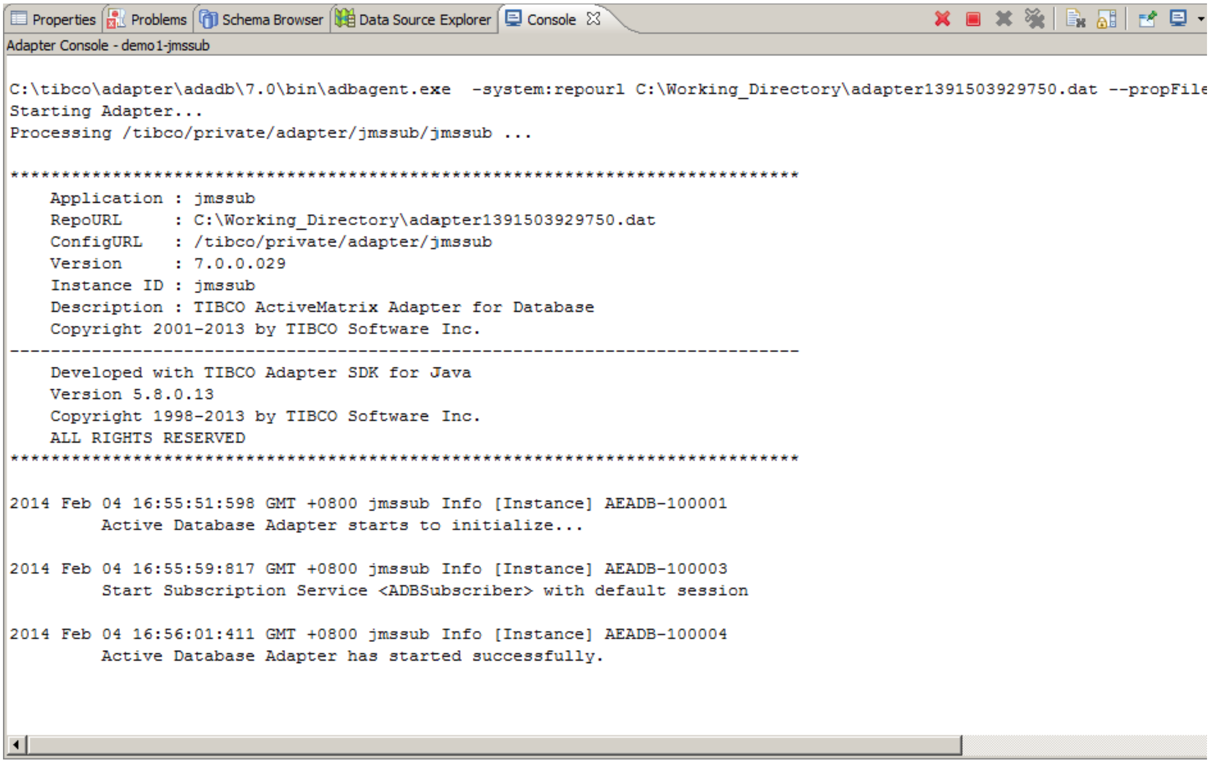
Task F Test the Adapter Configurations

Use the Adapter Launcher in TIBCO Business Studio to test the adapter configurations.

1. Start the EMS server: From the Windows Start menu, click **Run** and enter **`services.msc`**. In the Services dialog, start **TIBCO EMS Server**.

2. Start the subscriber adapter: Right-click the **`jmssub.adadbmodel`** adapter configuration in the Project Explorer, and from the pop-up menu, click **Run**

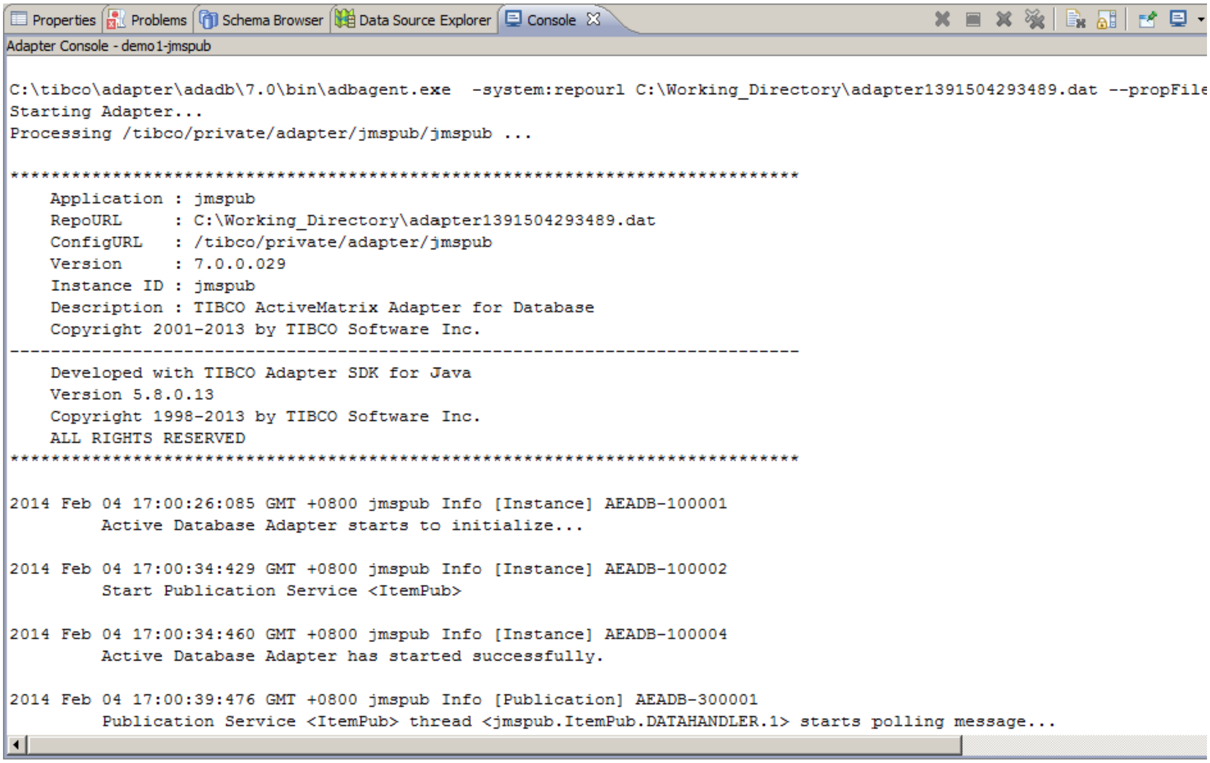
As > Adapter Launcher. The messages displayed in the Console view confirms the adapter is started successfully.



```
C:\tibco\adapter\adadb\7.0\bin\adbagent.exe  -system:repourl C:\Working_Directory\adapter1391503929750.dat --propFile
Starting Adapter...
Processing /tibco/private/adapter/jmssub/jmssub ...

*****************************************************************************
    Application : jmssub
    RepoURL     : C:\Working_Directory\adapter1391503929750.dat
    ConfigURL   : /tibco/private/adapter/jmssub
    Version     : 7.0.0.029
    Instance ID : jmssub
    Description : TIBCO ActiveMatrix Adapter for Database
    Copyright 2001-2013 by TIBCO Software Inc.
-----------------------------------------------------------------------------
    Developed with TIBCO Adapter SDK for Java
    Version 5.8.0.13
    Copyright 1998-2013 by TIBCO Software Inc.
    ALL RIGHTS RESERVED
*****************************************************************************

2014 Feb 04 16:55:51:598 GMT +0800 jmssub Info [Instance] AEADB-100001
         Active Database Adapter starts to initialize...

2014 Feb 04 16:55:59:817 GMT +0800 jmssub Info [Instance] AEADB-100003
         Start Subscription Service <ADBSubscriber> with default session

2014 Feb 04 16:56:01:411 GMT +0800 jmssub Info [Instance] AEADB-100004
         Active Database Adapter has started successfully.
```

3. Start the publisher adapter: Right-click the **`jmspub.adadbmodel`** adapter configuration in the Project Explorer, and from the pop-up menu, click **Run**

As > Adapter Launcher. The messages displayed in the Console view confirms the adapter is started successfully.



```
C:\tibco\adapter\adadb\7.0\bin\adbagent.exe  -system:repourl C:\Working_Directory\adapter1391504293489.dat --propFile
Starting Adapter...
Processing /tibco/private/adapter/jmspub/jmspub ...

*****************************************************************************
    Application : jmspub
    RepoURL     : C:\Working_Directory\adapter1391504293489.dat
    ConfigURL   : /tibco/private/adapter/jmspub
    Version     : 7.0.0.029
    Instance ID : jmspub
    Description : TIBCO ActiveMatrix Adapter for Database
    Copyright 2001-2013 by TIBCO Software Inc.
-----------------------------------------------------------------------------
    Developed with TIBCO Adapter SDK for Java
    Version 5.8.0.13
    Copyright 1998-2013 by TIBCO Software Inc.
    ALL RIGHTS RESERVED
*****************************************************************************

2014 Feb 04 17:00:26:085 GMT +0800 jmspub Info [Instance] AEADB-100001
         Active Database Adapter starts to initialize...

2014 Feb 04 17:00:34:429 GMT +0800 jmspub Info [Instance] AEADB-100002
         Start Publication Service <ItemPub>

2014 Feb 04 17:00:34:460 GMT +0800 jmspub Info [Instance] AEADB-100004
         Active Database Adapter has started successfully.

2014 Feb 04 17:00:39:476 GMT +0800 jmspub Info [Publication] AEADB-300001
         Publication Service <ItemPub> thread <jmspub.ItemPub.DATAHANDLER.1> starts polling message...
```

4. Insert some data into the source table by running the following SQL statements or the `demo1ins_ora.sql` script and commit the inserts:

   ```
   SQL> insert into ITEM_TABLE values(111,'Oak Table',499.95);
   SQL> commit;
   ```

   You can insert additional data rows if you need to. If you do, be aware that the first column (containing the value `111` in the example) is a primary key and must contain a value that is unique within the table.

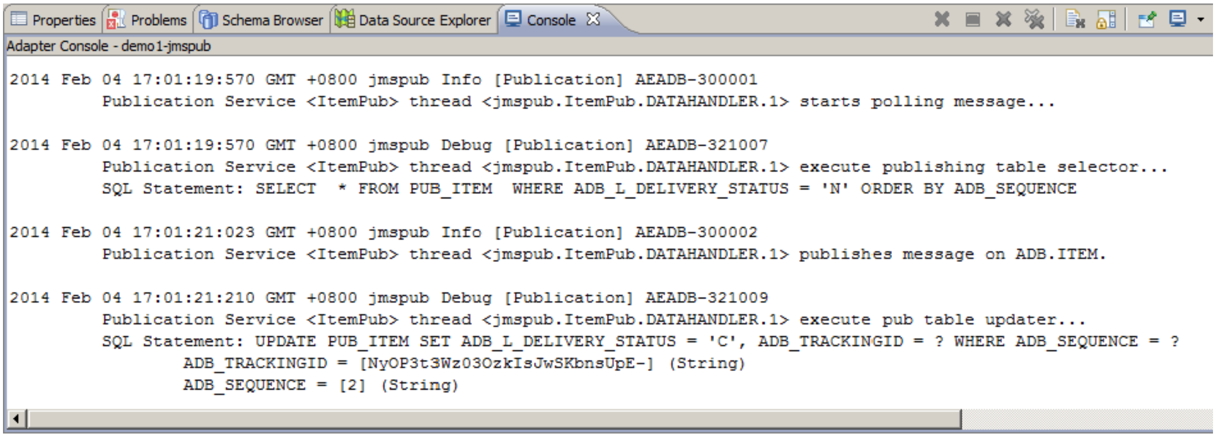
5. Verify the row in the source table has been inserted into the destination table by running the following query SQL:

```
SQL> select * from SUB_ITEM;
```

The following example result confirms that the data has been inserted:

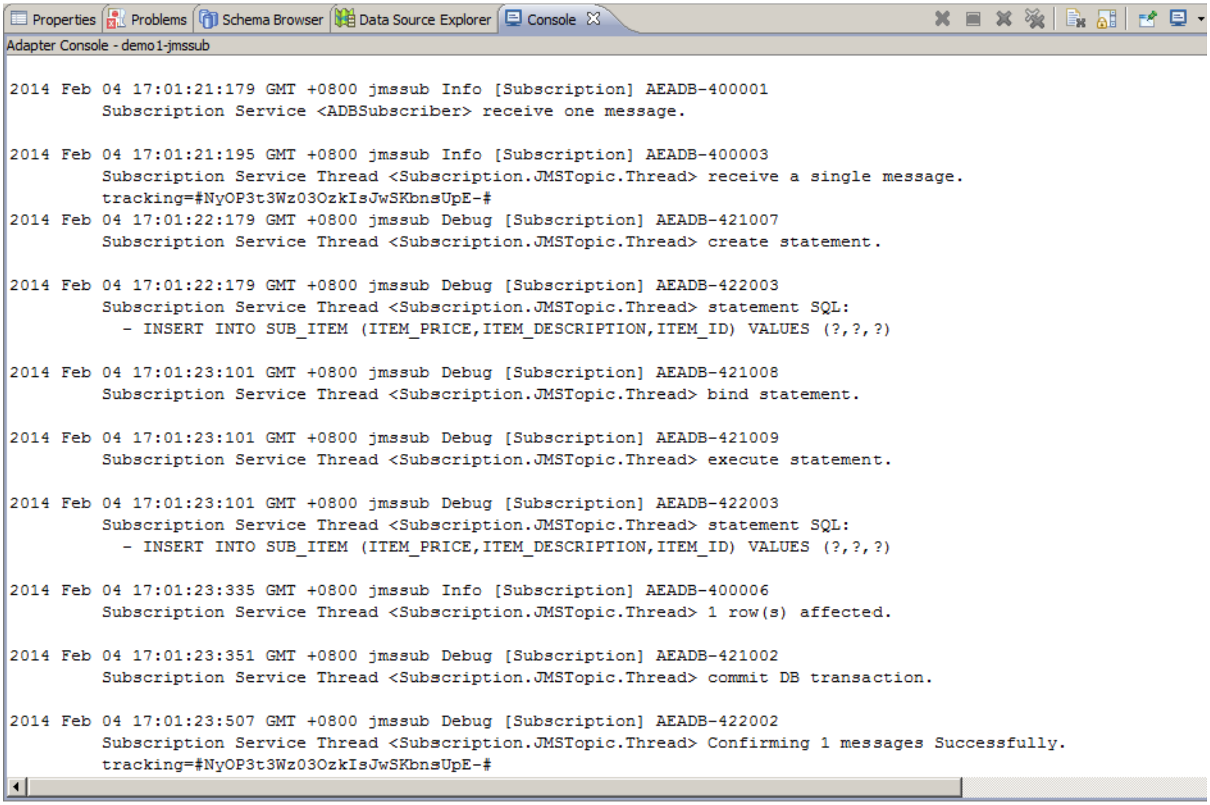
```
ITEM_ID
----------
ITEM_DESCRIPTION
----------------------------------------------------

ITEM_PRICE
-----------
   111
Oak Table

   499.95
```

6. Verify the status messages logged by the Publication Service:



```
2014 Feb 04 17:01:19:570 GMT +0800 jmspub Info [Publication] AEADB-300001
        Publication Service <ItemPub> thread <jmspub.ItemPub.DATAHANDLER.1> starts polling message...

2014 Feb 04 17:01:19:570 GMT +0800 jmspub Debug [Publication] AEADB-321007
        Publication Service <ItemPub> thread <jmspub.ItemPub.DATAHANDLER.1> execute publishing table selector...
        SQL Statement: SELECT  * FROM PUB_ITEM  WHERE ADB_L_DELIVERY_STATUS = 'N' ORDER BY ADB_SEQUENCE

2014 Feb 04 17:01:21:023 GMT +0800 jmspub Info [Publication] AEADB-300002
        Publication Service <ItemPub> thread <jmspub.ItemPub.DATAHANDLER.1> publishes message on ADB.ITEM.

2014 Feb 04 17:01:21:210 GMT +0800 jmspub Debug [Publication] AEADB-321009
        Publication Service <ItemPub> thread <jmspub.ItemPub.DATAHANDLER.1> execute pub table updater...
        SQL Statement: UPDATE PUB_ITEM SET ADB_L_DELIVERY_STATUS = 'C', ADB_TRACKINGID = ? WHERE ADB_SEQUENCE = ?
                ADB_TRACKINGID = [NyOP3t3Wz03OzkIsJwSKbnsUpE-] (String)
                ADB_SEQUENCE = [2] (String)
```

7. Verify the status messages logged by the Subscription Service:



```
2014 Feb 04 17:01:21:179 GMT +0800 jmssub Info [Subscription] AEADB-400001
         Subscription Service <ADBSubscriber> receive one message.

2014 Feb 04 17:01:21:195 GMT +0800 jmssub Info [Subscription] AEADB-400003
         Subscription Service Thread <Subscription.JMSTopic.Thread> receive a single message.
         tracking=#NyOP3t3Wz03OzkIsJwSKbnsUpE-#
2014 Feb 04 17:01:22:179 GMT +0800 jmssub Debug [Subscription] AEADB-421007
         Subscription Service Thread <Subscription.JMSTopic.Thread> create statement.

2014 Feb 04 17:01:22:179 GMT +0800 jmssub Debug [Subscription] AEADB-422003
         Subscription Service Thread <Subscription.JMSTopic.Thread> statement SQL:
           - INSERT INTO SUB_ITEM (ITEM_PRICE,ITEM_DESCRIPTION,ITEM_ID) VALUES (?,?,?)

2014 Feb 04 17:01:23:101 GMT +0800 jmssub Debug [Subscription] AEADB-421008
         Subscription Service Thread <Subscription.JMSTopic.Thread> bind statement.

2014 Feb 04 17:01:23:101 GMT +0800 jmssub Debug [Subscription] AEADB-421009
         Subscription Service Thread <Subscription.JMSTopic.Thread> execute statement.

2014 Feb 04 17:01:23:101 GMT +0800 jmssub Debug [Subscription] AEADB-422003
         Subscription Service Thread <Subscription.JMSTopic.Thread> statement SQL:
           - INSERT INTO SUB_ITEM (ITEM_PRICE,ITEM_DESCRIPTION,ITEM_ID) VALUES (?,?,?)

2014 Feb 04 17:01:23:335 GMT +0800 jmssub Info [Subscription] AEADB-400006
         Subscription Service Thread <Subscription.JMSTopic.Thread> 1 row(s) affected.

2014 Feb 04 17:01:23:351 GMT +0800 jmssub Debug [Subscription] AEADB-421002
         Subscription Service Thread <Subscription.JMSTopic.Thread> commit DB transaction.

2014 Feb 04 17:01:23:507 GMT +0800 jmssub Debug [Subscription] AEADB-422002
         Subscription Service Thread <Subscription.JMSTopic.Thread> Confirming 1 messages Successfully.
         tracking=#NyOP3t3Wz03OzkIsJwSKbnsUpE-#
```

8. Stop both the subscriber and publisher adapters: click the stop button on the top left of the Console view of each started adapter.

9. Exit TIBCO Business Studio.

10. Stop the EMS server: From the Windows Start menu, click **Run** and enter **`services.msc`**. In the Services dialog, stop **TIBCO EMS Server**.

Task G Clean up the Database

This cleanup script removes the example tables that were created by the `demo1_cleanup_`*`databasevendor`*`.sql` script.

1. In a command window, change directory to the `demo1` directory. For example:

   `>` **`cd`** *TIB_ADADB_HOME*\**`demo\demo1`**

2. Execute the appropriate `demo_cleanup.sql` script in the subdirectory.

   `>` **`sqlplus`** *userid/password@dbService* **`@demo1_cleanup_`*****`databasevendor`*****`.sql`**

For example:

```
> sqlplus karlh/karlh@ORCL @demo1_cleanup_ora.sql
```

Chapter 3 Working with Adapter Configurations and Services

For an adapter, you need to configure the adapter itself, the adapter services, and all the resources required by the adapter. The configured services can then be wired to a business process through associated activities in the Adapter palette.

Topics

Overview of Adapter Configuration

The configuration of an adapter in a project contains the configuration of the adapter itself, the adapter services, and all the resources required by the runtime adapter.

Working with adapter configurations includes the following tasks:

1. Create an Adapter Configuration and Adapter Services

   The adapter user interface provides a series of flexible approaches to create adapter configurations and services. See Workflows with Adapter Configurations and Services on page 33.

2. Configure an adapter configuration and its services:

   Each adapter configuration must have its corresponding resources configured. The Adapter Configuration editor is a form based editor split into several tabs. Each tab corresponds to one type of resource and has one or more configuration sections that are presented in panels. Use these tabs to configure the adapter resource.

   The Project Explorer view shows a virtual outline of the adapter configuration. The outline depicts the virtual containment hierarchy of the configuration. Each node in the hierarchy is an Eclipse resource that is backed by a physical file. In most cases, a resource node in the outline maps to a tab in the Adapter Configuration editor.

   You can first set the preferred configuration settings using the Preferences dialog. Then configure each resource by using the tabs in the editor. See the sections that correspond to each resource for details. See Configuring Adapter Preferences on page 11 for preferences configuration.

Many of the configuration options make use of module properties. See Configuring Module Properties on page 12 for details on how to define and use module properties.

3. Validate an Adapter Configuration

   As you work with resources in TIBCO Business Studio, errors and warnings are logged automatically in the Problems view. Fix the errors before you test the adapter configuration. See Validating an Adapter Configuration on page 71.

4. Test an Adapter Configuration

   The Adapter Launcher is a tool for testing an adapter configuration. See Testing an Adapter Configuration on page 72 for the test procedure.

Workflows with Adapter Configurations and Services

The adapter user interface provides flexible ways to create adapter configurations and services.

Creating Adapter Configurations

An *adapter configuration* in a project contains all information needed by the runtime adapter.

Before creating an adapter configuration, you need to first prepare the design environment. See Preparing the Development Environment on page 10 for details.

To create an adapter configuration:

1. Either click **File > New > Other**, or right-click the project folder in the Project Explorer and from the pop-up menu, click **New > Other**.
2. In the **New** dialog, specify the search filter **adapter**. Then select **TIBCO Adapters > Adapter for Database > Adapter for Database Configuration** and click **Next**.
3. In the Adapter for Database Configuration wizard, specify the name for the adapter configuration.
4. Click **Finish**.

The created adapter configuration is stored in the new *adapter_configuration_name*`.adadbmodel` file.

The Adapter Configuration folder is created along with the creation of the first Adapter Configuration.

Creating Adapter Services

TIBCO Adapters exchange data with sources and target applications by using a service architecture. *Adapter services* are abstractions that describe how adapters work together with other applications.

You can create an adapter service in one of the following ways:

- Create an Adapter Service from the Project Explorer View, page 34
- Create an Adapter Service from the Adapter Services tab, page 34

Create an Adapter Service from the Project Explorer View

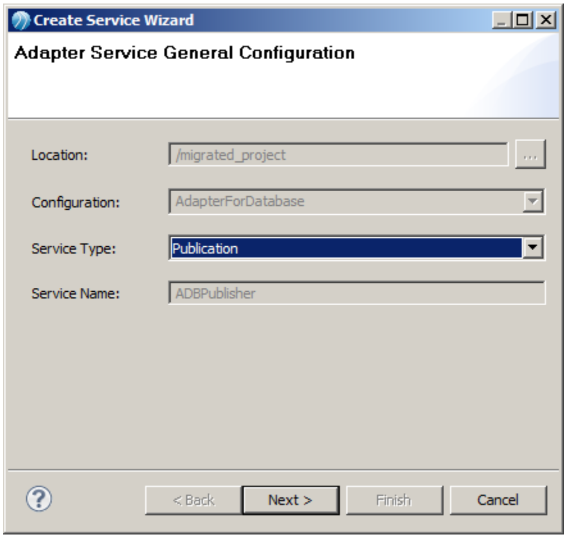
Right-click the **Adapter Services** folder of the adapter configuration in the Project Explorer view and from the pop-up menu, click **New Service**. The Create Service wizard is displayed. Using the wizard, you can create an adapter service. See Using the Create Service Wizard on page 34 for details.

Create an Adapter Service from the Adapter Services tab

the Adapter Services tab is located in the adapter configuration editor. In the All Adapter Services panel, either click **Add**, or right-click the service list pane and from the pop-up menu, click **New Service**. The Create Service wizard will open. Using the wizard, you can create an adapter service. See Using the Create Service Wizard on page 34 for details.

Using the Create Service Wizard

The following figure shows the Create Service Wizard.



Create Service Wizard

Adapter Service General Configuration

Location: /migrated_project

Configuration: AdapterForDatabase

Service Type: Publication

Service Name: ADBPublisher

When using wizard to create an adapter service, follow these steps:

Information that the Create Service Wizard provides is inherited from the adapter preferences. To maintain the information stored in the Preferences dialog, see Configuring Adapter Preferences on page 11.

1. From the Service Type list, select the service type you want to create. The adapter configuration and service name are filled automatically.

   — For Publication Services, Subscription Services, or Request-Response Services in Custom RPC mode, click **Next** to pick a table or stored procedure.

   — For Request-Response Services in Standard RPC mode or Request-Reply mode, click **Next** if you want to select a transport session or **Finish**.

2. In the Schema Type page, you can choose a table for Publication Services and Subscription Services, or a stored procedure for Request-Response Services in Custom RPC mode. For Request-Response Services in Standard RPC mode or Request-Reply mode, click **Next** to select a transport session or **Finish**.

   Follow these steps to choose a table or stored procedure:

   a. Click the **...** button to open the ADB Business Object Schema Picker dialog

   b. Click the tab and select the table or stored procedure:

   — **Local Business Object** If you want to choose an object from the local repository, use this tab. In the tab, pick the fetched object you want to use. Click **OK**.

   — **Remote Business Object** If you want to choose an object that has not been fetched to the local repository, use this tab. Create a connection for the database, if it does not exist. Then right-click the connection and fetch tables or stored procedures by following the same task sequence as described in Fetching Tables or Stored Procedures on page 80. Click **OK**.

Depending on the service type you choose, the schema objects displayed are either tables for Publication Services and Subscription Services, or procedures for Request-Response Services.

3. (Optional) In the Transport Session page, choose a transport session for the service:

   a. Click the **...** button to open the Transport Selection dialog.

   b. If the transport session you want to use has already been created, select it from the list.

   c. If you want to create a new transport session, click **New**. Fill in the information in the New Transport dialog, and click **OK**. Then select the session you just created.

   d. Click **OK**.

4. Click **Finish**.

Configuring an Adapter Configuration

You can specify the general information and crucial options for an adapter configuration using the Configuration tab. The information is grouped in panels.

- Configuring the General Information of the Adapter Configuration, page 36
- Configuring the Database Connection, page 37
- Getting Started, page 40

Configuring the General Information of the Adapter Configuration

You can specify the name and description of the adapter configuration.

Table 5 General Information

Name	Description
Adapter Name	Type of the adapter configuration.
Instance Id	ID of the adapter configuration. The value of this field comes from the file name you specified when you create the adapter configuration. If you change this field and save the configuration, the adapter configuration file name will also be changed. See Guidelines for Choosing an Instance ID on page 36 for more information.
Description	(Optional) Description of the adapter configuration.
Message Filter	Not used.

Guidelines for Choosing an Instance ID

- An instance ID must use alphanumeric characters. You can use underscore (_) characters, but you cannot contain any spaces. The entire instance name must be less than 80 characters.
- An instance ID cannot use module properties.
- An instance ID must be unique in the whole project.

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

Configuring the Database Connection

In the Database Configuration panel, you need to specify the settings of the connection between the adapter and the database.

Table 6 Database Configuration

Name	Description
Vendor	Select the database vendor to which the adapter connects. In the list, vendor names enclosed in parenthesis are not supported. For example, Informix and INGRES.
DB2 AS400 Library	Appears when DB2 AS400 is selected in the Vendor list. A DB2 AS400 library name is required. By default, this library name is same as the AlternateID specified in the JDBC URL field in the Design-time tab. Other database libraries are also supported. **Note**: • You must configure the DB2 AS400 Library field in the Configuration Tab, otherwise the publishing table cannot be created. • You must also set the trigger option in the DB2 AS400 Options tab of the Publication Services.
Convert Number Datatype to String	Displayed when Oracle is selected in the Vendor field. When this option is selected, the adapter will use string to represent the number data type at runtime.
Write to Database on Save	The default mode if you want to write configuration settings to the database when this project is saved.

Table 6 Database Configuration (Cont'd)

Name	Description
Connection Reference	The database connection resource used by the adapter configuration. To specify a database connection reference, click the **...** browse button next to the Connection Reference field and choose a reference from the displayed connection configuration list in the opened dialog. If the list does not include the database connection you want to use, click **New** to open the New Connection dialog and add one. See Add a Database Connection Resource on page 38 for how to add a connection.
Link Connection to Schema Browser	Click this link and a new destination whose name and database connection information is the same as the specified connection reference will be created in the Schema Browser. If a destination with the same name and information as the specified connection reference already exists in the Schema Browser, the Schema Browser is opened for you. If their database connection information is not identical, a warning message will be displayed.

Add a Database Connection Resource

You need to specify the fields in the New Connection dialog. The following table explains the connection configuration fields.

Table 7 Connection Configuration

Name	Description
Connection Resource	Name of the database resource.
JDBC Driver	Name and URL of the JDBC driver used during configuration. Table 8 lists all the supported JDBC drivers and the associated URLs. For detailed parameter descriptions, see your JDBC driver documentation.
JDBC URL	
User Name	Database user that the adapter uses to connect to the database.
Password	Password for the database user.
Default Schema	Schema name for the current user. Usually it is the same as the database name. But it can be different for some specific databases. For example, when using a Microsoft SQL Server database, the default schema is DBO. When using a PostgreSQL database, the default schema is Public.

Table 7 Connection Configuration (Cont'd)

Name	Description
Test Connection	Click this button to verify the connection parameters specified. A popped-up dialog will display if the connection is successful or failed.

Table 8 JDBC Drivers and URLs

Database	Driver and URL
Oracle	**JDBC Driver**: `tibcosoftwareinc.jdbc.oracle.OracleDriver` **JDBC URL**: `jdbc:tibcosoftwareinc:oracle://`*server_name*`:1521;SID=ORCL` **Note**: The default port number is `1521`.
Microsoft SQL Server	**JDBC Driver**: `tibcosoftwareinc.jdbc.sqlserver.SQLServerDriver` **JDBC URL**: `jdbc:tibcosoftwareinc:sqlserver://`*server_name*`:1433;databaseName=`*database_name* **Note**: The default port number is `1433`.
Sybase	**JDBC Driver**: `tibcosoftwareinc.jdbc.sybase.SybaseDriver` **JDBC URL**: `jdbc:tibcosoftwareinc:sybase://`*server_name*`:5000;DatabaseName=`*database_name*
DB2 OS390	**JDBC Driver**: `tibcosoftwareinc.jdbc.db2.DB2Driver` **JDBC URL**: `jdbc:tibcosoftwareinc:db2://`*server_name*`:`*port*`;locationName=`*location_name*`;packageName=`*packageName*
DB2 AS400	**JDBC Driver**: `tibcosoftwareinc.jdbc.db2.DB2Driver` **JDBC URL**: `jdbc:tibcosoftwareinc:db2://`*server_name*`:`*port*`;locationName=`*location_name*`;AlternateID=`*library* **Note:** The publishing table cannot be created if `AlternateID=`*library* is not in JDBC URL.
DB2 UDB	**JDBC Driver**: `tibcosoftwareinc.jdbc.db2.DB2Driver` **JDBC URL**: `jdbc:tibcosoftwareinc:db2://`*server_name*`:50000;databaseName=`*database_name*`;packageName=DEF00`

Table 8 JDBC Drivers and URLs (Cont'd)

Database	Driver and URL (Cont'd)
MySQL	**JDBC Driver**: `tibcosoftwareinc.jdbc.mysql.MySQLDriver` **JDBC URL**: `jdbc:tibcosoftwareinc:mysql://`*server_name*`:3306/DatabaseName=`*database_name*
Teradata	**JDBC Driver**: `com.teradata.jdbc.TeraDriver` **JDBC URL**: `jdbc:teradata://`*server_name*`/database=`*database_name*
PostgreSQL	**JDBC Driver**: `org.postgresql.Driver` **JDBC URL**: `jdbc:postgresql://`*server_name*:*port*`/`*database_name*

Getting Started

This panel lists the **Configure Adapter Services** link. Click this link, you will be directed to the Adapter Services tab. See Configuring Adapter Services on page 41 for details.

Configuring Adapter Services

You can configure the adapter services for an adapter configuration using the Adapter Services tab.

You need to specify the following configuration for each service:

- General information: For each service, you need to specify the general information of the service and the transport used by the service. See Configuring the General Information of Adapter Services on page 41.
- Schema information and service specific options: Each service must be associated with the tables or stored procedures that constrains the data communicated or operations executed by the service. Also, each service has its specific options that need to be configured.

  See the corresponding chapter of each service for details:
- Chapter 7, Configuring Publication Services, page 101
- Chapter 8, Configuring Subscription Services, page 141
- Chapter 9, Configuring Request-Response Services, page 159

Configuring the General Information of Adapter Services

The general information of a service includes:

- Service name and description:

  See Configure the Name and Description on page 42 for more information.
- Transport options: The transport options includes the transport to use, destination or subject, and wire format. It also includes the session and endpoint used by the adapter service.

  When you add a service to an adapter, the adapter user interface automatically creates the corresponding session and endpoint, depending on the transport protocol and delivery mode being used. Both session and endpoint are concepts in TIBCO Adapter SDK.

  See Configure the Transport Options on page 42 for more information.
- Class reference: The class reference points to the AE Schema used by the service.

  When you create a service with a schema object, the adapter user interface transforms the schema information into an AE Schema. Different from database schema, an AE Schema is recognized by TIBCO Adapter SDK.

See Configure the Class Reference on page 45 for more information.

Configure the Name and Description

You can configure the service name and description.

Table 9 Adapter Service - Configuration Tab - Configuration Panel

Name	Description
Name	Name of the adapter service.
Description	(Optional) Description of the adapter service.

Configure the Transport Options

You can change and configure the transport options using the Transport panel.

The transport options include the following two parts:

- Session and endpoint

  Endpoints represent the services that an adapter provides. *Sessions* encapsulate transport information. A service encapsulates both an endpoint and the corresponding session.

  When creating an adapter service, a corresponding endpoint and session are created automatically. In general, you do not need to change these defaults. If you want to change the session, click the **...** browse button next to the field to open the Model Select dialog and select the session you want to use. You can also add a session by clicking **New** in the dialog. This will open the same New Transport dialog as you click **Add** in the Transports tab. See Configuring Transports on page 46 for supported sessions and endpoints and their configuration options.

  To open the specific session or endpoint in the Transports tab, click the **Session Reference** or **Endpoint Reference** link.

- Transport subject or destination, and wire format

  Two groups of settings are applicable, depending on the type of the transport session you choose:

  — Transport Type: JMS, page 43

  — Transport Type: Rendezvous, page 43

Transport Type: JMS

This section introduces the options for the JMS transport session.

Table 10 Service - Configuration Tab - Transport Panel: JMS

Name	Description
Destination	Name on which the service publishes to a topic or sends messages to a queue.
Wire Format	Format in which messages are sent. Available option is: • XML Message See Guideline for Configuring the Wire Format on page 44 for details.
Delivery Mode (JMS Topic Connection only)	Delivery mode for the message. Available options for Publication Services are: • Non-Persistent • Persistent Available options for Subscription Services and Request-Response Services are: • Non-Durable • Durable See Guideline for Configuring the Delivery Mode (JMS Only) on page 45 for details.

Transport Type: Rendezvous

This section introduces the options for the Rendezvous transport session.

Table 11 Service - Configuration Tab - Transport Panel: Rendezvous

Field	Description
Subject	Subject name to use by default when publishing.

Table 11 Service - Configuration Tab - Transport Panel: Rendezvous (Cont'd)

Field	Description
Wire Format	Format in which messages are sent. Available options are: • ActiveEnterprise Message • Rendezvous Message • XML Message See Guideline for Configuring the Wire Format on page 44 for details.

Guideline for Configuring the Wire Format

Wire Format indicates the format in which messages are to be sent or received. Services must use the same wire format to exchange data.

> Adapter services must use the same transport type and wire format to exchange data.

Three wire formats are available:

- ActiveEnterprise Message (TIBCO Rendezvous only)

  ActiveEnterprise Message is an externally-described XML message format supported by TIBCO Adapter SDK. ActiveEnterprise standard wire format provides class information and packing rules for the TIBCO Adapter SDK set of data types. This format allows ActiveEnterprise components to perform extra validation on messages sent or received. Control information for validation is sent in the message. If no control information is included, an exception is returned to the subscriber.

  See *TIBCO Adapter SDK Programmer's Guide* for details about the control information generated and sent with TIBCO ActiveEnterprise messages.

- Rendezvous Message (TIBCO Rendezvous only)

  Rendezvous Message is a self-describing message format used by TIBCO Rendezvous applications. Control information for validation is *not* sent in the message.

- XML Message

  XML Message conforms to specifically constructed and fully compliant XML Schema (XSD) based on the existing definition of the ActiveEnterprise schema.

Guideline for Configuring the Delivery Mode (JMS Only)

For JMS transport, different delivery modes are available for different services:

- For Publication Services, the following delivery modes are supported:
  - — **Persistent** The message will be available to a JMS client even if the JMS server goes down.
  - — **Non-persistent** The message will not be available to a JMS client if the JMS server goes down.
- For Subscription Services and Request-Response Services, the Delivery Mode option is available only when the Connection Factory Type is Topic. See Connection Factory on page 52 for details about connection factory types.

  The following delivery modes are supported:
  - — **Durable** Using this delivery mode, an adapter service will receive the messages sent to it when it is not currently running. The service is registered with the JMS server. The JMS server stores messages sent to the service even when the service is down. When the service has recovered, it will receive the stored messages.
  - — **Non-durable** Using this delivery mode, an adapter service will not receive the messages sent to it when it is not running. The service is not registered with the JMS server. The JMS server does not hold messages sent to a non-durable service when the service is down.

  See *TIBCO Enterprise Message Service User's Guide* for more information about delivery modes.

Configure the Class Reference

The Schema panel displays the associated AE Schema for an adapter service.

To open the specific AESchema file, click the **Class Reference** link. The file is opened in the editor with the Classes tab displayed.

When you create a service with a schema object, the adapter user interface transforms the schema information into an AE Schema. Different from database schema, an AE Schema is recognized by TIBCO Adapter SDK. See Difference between Database Schema and TIBCO ActiveEnterprise Schema on page 76 for more details.

Configuring Transports

The primary task of an adapter is to retrieve or send data. When you add an adapter service to an adapter, the adapter user interface automatically creates the corresponding session and endpoint to encapsulate transport information necessary for data communication, depending on the transport protocol and delivery mode being used.

Creating sessions explicitly and adding endpoints to the session is not recommended. Instead, you should create the service you need, and let the adapter user interface create the session and endpoint. If there are changes you cannot make directly to the service, you can then make them to the corresponding session or endpoint by using the Transport tab. The information is grouped in panels.

Sessions and Endpoints

Both sessions and endpoints are concepts in TIBCO Adapter SDK, the fundamental class library used in the adapter implementation.

Endpoints send or receive the data. They represent the service an adapter provides. Each endpoint is associated with a *session* that is used to communicate with the source or target application. A session encapsulates the transport information of an adapter service.

See *TIBCO Adapter SDK Programmer's Guide* for more information about these concepts.

Supported Endpoints

Publishers, Subscribers, Clients, or Servers are the endpoints that are available in an adapter. An adapter service encapsulates both an endpoint and the corresponding session. The services TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) supports include:

- Publication Service—a Publisher endpoint and associated session
- Subscription Service—a Subscriber endpoint and associated session
- Request-Response Service—a Server endpoint and associated session

Supported Sessions

Both TIBCO Rendezvous and JMS transport types are supported:

- For TIBCO Rendezvous transports, sessions of the following kinds of Quality of Service are available:

  — **Reliable** (RV)

  Reliable Message Delivery ensures that each multicast or broadcast message is received as long as the physical network and packet recipients are working. It also ensures that the loss of a message is detected.

  Reliable Message Delivery can compensate for brief network failures, because it can retransmit a message on request if the first attempt fails. This option is appropriate when message delivery is expected but some loss can be tolerated. When this Quality of Service is chosen, an RV session will be used.

  — **Certified** (RVCM)

  Certified Message Delivery guarantees that every certified message reaches its intended recipient in the order sent. A message can be sent across network boundaries, and if a network fails, delivery attempts continue until delivery succeeds or until the time limit of the message expires. This is often called guaranteed delivery. When this Quality of Service is chosen, an RVCM session will be used.

  — **Distributed Queue (Subscription Service and Request-Response Service only)** (RVCMQ)

  Distributed Queue delivers a message to one of many service listeners (workers). It contains features of both Certified Messaging and Fault Tolerance.

  See *TIBCO Rendezvous Concepts* for more information about the available TIBCO Rendezvous types of Quality of Service.

- For JMS transports, sessions of the following standard connection factory types are available:

  — **Topic**

  A message published to a topic is broadcast to one or more subscribers. All messages published to the topic are received by all services that have subscribed to the topic. This messaging model is known as publish-subscribe.

  — **Queue**

  A message sent to a queue is consumed by one and only one receiver. Each message has only one receiver, though multiple receivers may connect to the queue. The first receiver to access the queue gets the message. The other receivers do not. This messaging model is known as point-to-point.

See *TIBCO Enterprise Message Service User's Guide* for more information about connection factories.

Adding Sessions and Endpoints

Use the All Adapter Transports panel to add or remove a session and its endpoints.

Configuring Sessions

Select the session you added from the All Adapter Transports panel, the configuration options for the session are displayed in the Configuration panel. The transport type of the session you select determines session specific options you need to configure.

- Rendezvous Sessions, page 48
- JMS Sessions, page 51

Specify the following options:

- **General Options** No matter which transport type of the session you select, JMS or Rendezvous, the general information of the session is displayed in a generic configuration panel at the top of the Configuration panel.
- **Advanced Options** In addition to the generic configuration, some sessions can have more configuration options that need to be specified. These options are displayed in one or more configuration panels.

Rendezvous Sessions

You can specify general information using the Rendezvous Configuration panel. For options specific to the session type, use the Rendezvous Options panel. This section introduces:

- Rendezvous Configuration Reference, page 49
- Rendezvous Options—Reliable Session Reference, page 49
- Rendezvous Options—Certified Session Reference, page 49
- Rendezvous Options—Distributed Queue Session Reference, page 50

See the TIBCO Rendezvous documentation for more information.

Rendezvous Configuration Reference

The following table lists the options in the Rendezvous Configuration panel.

Table 12 Rendezvous Configuration Reference

Name	Description
Name	Name of the TIBCO Rendezvous transport.
Description	(Optional) Description of the transport.
Daemon	Module property that specifies the TIBCO Rendezvous daemon for this session.
Network	Module property that specifies the network for this transport. By default, the property is an empty string, which is interpreted as the primary network. Using this attribute only makes sense on computers with more than one network interface.
Service	Module property that specifies the service for this transport. By default, the property is defined to be the default TIBCO Rendezvous service (7500).
Connection Type	The type of the connection: • Reliable • Certified • Distributed Queue

Rendezvous Options—Reliable Session Reference

No additional configuration is required for Reliable sessions.

Rendezvous Options—Certified Session Reference

The following table lists the options for Certified sessions.

Table 13 Rendezvous Options—Certified Session Reference

Name	Description
CM Name	Used to identify the delivery tracking session. It must be unique across the entire network.
Ledger File	Module property that points to a ledger file. If the value of the variable value is a valid file name, the transport uses a file-based ledger.

Table 13 Rendezvous Options—Certified Session Reference (Cont'd)

Name	Description
Sync Ledger File	This check box controls the behavior when updating the ledge file. • **Selected** Operations that update the ledger file do not return until the changes are written to the storage medium. • **Cleared** The operating system writes changes to the storage medium asynchronously.
Relay Agent	Relay agent for this transport.
Require Old Message	This check box indicates whether a persistent correspondent requires delivery of messages sent to a previous transport with the same name for which delivery was not confirmed. Its value affects the behavior of other delivery-tracking senders. • **Selected** If the name attribute is non-NULL, then this transport requires certified senders to retain unacknowledged messages sent to this persistent correspondent. • **Cleared** Messages are not retained.
Message Timeout (sec)	Maximum time (in seconds) that this call can block while waiting for a reply.

Rendezvous Options—Distributed Queue Session Reference

The following table lists the options for Distributed Queue sessions.

Table 14 Rendezvous Options—Distributed Queue Session Reference

Name	Description
CMQ Name	Sequence of the module property that specifies the name of the queue.
Worker Weight	Relative worker weights assist the scheduler in assigning tasks. When the scheduler receives a task, it assigns the task to the available listener with the greatest listener weight. Default is 0.
Worker Tasks	Worker tasks for this session. Default is 0.
Worker Complete Time (sec)	If the complete time is non-zero, the scheduler waits for a worker member to complete an assigned task. If the complete time elapses before the scheduler receives completion from the worker member, the scheduler reassigns the task to another worker member. Default is 0.

Table 14 Rendezvous Options—Distributed Queue Session Reference (Cont'd)

Name	Description
Scheduler Weight	Represents the ability of this session to fulfill the role of scheduler, relative to other members of the same queue. The queue members use relative scheduler weight values to elect one member as the scheduler. Members with higher scheduler weight take precedence. Acceptable values range from 0 to 65545. Default is 0.
Scheduler Heartbeat (sec)	The scheduler session sends heartbeat messages at this interval (in seconds). All member session in the queue must specify the same value for this parameter. Acceptable values are the unsigned decimals. Default is 0.0.
Scheduler Activation (sec)	When the heartbeat signal from the scheduler has been silent for this interval (in seconds), the queue member with the greatest scheduler weight takes its place as the new scheduler. All member sessions in the queue must specify the same value for this parameter. Acceptable values are unsigned decimals. Defaults is 0.0.

JMS Sessions

You can specify the general configuration information using the JMS Configuration panel. For the session specific options, use the More Options panel.

This section introduces:

- JMS Configuration Reference, page 51
- More Options Reference, page 52

See the TIBCO Enterprise Message Service documentation for more information.

JMS Configuration Reference

The following table lists the options in the JMS Configuration panel.

Table 15 JMS Configuration Reference

Name	Description
Name	Name of the JMS transport.
Description	(Optional) Description of the transport.

Table 15 JMS Configuration Reference (Cont'd)

Name	Description
Connection Type	Lists the available connection types: • **Direct** (Default) The connection is direct. • **JNDI** A JNDI Server is used.
Provider URL	(Direct connection type only) URL of the server.
Connection Factory	(Direct connection type only) Two connection factory types are available: • **TopicConnectionFactory** A message published to a topic is broadcast to one or more subscribers. All messages published to the topic are received by all services that have subscribed to the topic. This messaging model is known as *publish-subscribe*. • **QueueConnectionFactory** A message sent to a queue is consumed by one and only one receiver. Each message has only one receiver, though multiple receivers may connect to the queue. The first receiver to access the queue gets the message. The other receivers do not. This messaging model is known as *point-to-point*.
JNDI Reference	(JNDI connection type only) JNDI server information.

You need to configure the JMSConnection parameters in JMSSharedResources (in the Project Resources folder) when the TIBCO ActiveMatrix BusinessWorks activities using JMS transport in TIBCO Business Studio. The JMSConnection parameters are configured in the **Transport** tab in TIBCO Designer.

More Options Reference

The following table lists the options in the More Options panel.

Table 16 More Options Reference

Name	Description
Client ID	ID of the client.
User Identity	Detailed information of the user.

Configuring Endpoints

Select the endpoint you added from the All Adapter Transports panel, the configuration options for the endpoint are displayed in the Configuration panel. The session type of the endpoint you select determines endpoint specific options you need to configure.

Publisher Endpoints

A Publisher sends data to TIBCO Rendezvous or TIBCO Enterprise Message Service. only Reliable sessions and Certified sessions can be associated with a Publisher. The session for a Publisher determines the endpoint options you need to specify.

TIBCO Rendezvous Publisher Endpoint Reference

You can add a Publisher endpoint for a TIBCO Rendezvous Reliable session or Certified session.

Table 17 TIBCO Rendezvous Publisher Endpoint Reference

Name	Description
Name	Name of this Publisher.
Description	Optional description of the Publisher.
Endpoint Type	Type for the current Publisher. • For a TIBCO Rendezvous Reliable Publisher, the default value is Rv Publisher. • For a TIBCO Rendezvous Certified Publisher, the default value is RvCm Publisher.
Wire Format	Format in which messages are sent. Available options are: • ActiveEnterprise Message (Default) • Rendezvous Message • XML Message See Guideline for Configuring the Wire Format on page 44 for details.
Subject	Subject with which this Publisher will send out messages.
Reply Subject	Reply subject for this Publisher.

Table 17 TIBCO Rendezvous Publisher Endpoint Reference (Cont'd)

Name	Description
Message Timeout (ms)	(Publishers for TIBCO Rendezvous Certified Sessions only) Time after which the message is discarded from the ledger file. The default value is 0 seconds, meaning that the timeout is infinite.
Pre-registered Listeners	(Publishers for TIBCO Rendezvous Certified Sessions only) Comma-separated list of listeners preregistered for this publisher. Refer to each listener using the `CmName` of the session.

JMS Publisher Endpoint Reference

You can add a Publisher endpoint for a JMS Topic or Queue session.

Table 18 TIBCO JMS Publisher Endpoint Reference

Name	Description
Name	Name of this Publisher.
Description	Optional description of the Publisher.
Endpoint Type	Type for the current Publisher. The only available value is Jms Publisher.
Delivery Mode	Delivery mode for the messages. Available options are: • Non-Persistent • Persistent (Default) See Guideline for Configuring the Delivery Mode (JMS Only) on page 45 for details.
Destination	Destination with which this Publisher will send out messages.
Reply destination	Reply destination for this Publisher.
Message Priority	Priority of the messages to send, as a value in the range [0,9]. The default value is 4.

Table 18 TIBCO JMS Publisher Endpoint Reference (Cont'd)

Name	Description
isCompressed	Select this check box if you want to compress the body of a message before sending the message to the server. Setting compression ensures that messages will take less memory space in storage.
Message Timeout (ms)	Time after which the message is discarded from the ledger file. The default value is 0 seconds, meaning that the timeout is infinite.

Subscriber Endpoints

A Subscriber specifies the data consumers in the applications.

TIBCO Rendezvous Subscriber Endpoint Reference

You can add a Subscriber endpoint for a TIBCO Rendezvous Reliable, Certified, or Distributed Queue session.

Table 19 TIBCO Rendezvous Subscriber Endpoint Reference

Name	Description
Name	Name of this Subscriber.
Description	Optional description of the Subscriber.
Endpoint Type	Type for the current Subscriber. • For a TIBCO Rendezvous Reliable Subscriber, the default value is Rv Subscriber. • For a TIBCO Rendezvous Certified Subscriber, the default value is RvCm Subscriber. • For a TIBCO Rendezvous Distributed Queue Subscriber, the default value is RvCmq Subscriber.
Startup State	State when starting up the endpoint. Available options are: • Active (Default) • Inactive

Table 19 TIBCO Rendezvous Subscriber Endpoint Reference (Cont'd)

Name	Description
Wire Format	Format in which messages are sent. Available options are: • ActiveEnterprise Message • Rendezvous Message (Default) • XML Message See Guideline for Configuring the Wire Format on page 44 for details.
Subject	Subject with which this Subscriber will receive messages.
Listen Timeout (ms)	If no message is received after this amount of time, the adapter performs any actions specified in the program for that case. The default value is 0.

JMS Subscriber Endpoint Reference

You can add a Subscriber endpoint for a JMS Topic or Queue session.

Table 20 TIBCO JMS Subscriber Endpoint Reference

Name	Description
Name	Name of this Subscriber.
Description	Optional description of the Subscriber.
Endpoint Type	Type for the current Subscriber. The only available value is Jms Subscriber.
Auto Confirm	Select this check box to let TIBCO Adapter SDK automatically confirms events for this Subscriber.
Destination	Destination with which this Subscriber will receive messages.
Message Selector	A message selector is a string that lets a client program specify a set of messages, based on the values of message headers and properties. A selector matches a message if, after substituting header and property values from the message into the selector string, the string evaluates to `true`. Consumers can request that the server deliver only those messages that match a selector.

Table 20 TIBCO JMS Subscriber Endpoint Reference (Cont'd)

Name	Description
Delivery Mode	(Subscribers for JMS Topic Sessions only) Delivery mode for the messages. Available options are: • Durable (Default) • Non-Durable See Guideline for Configuring the Delivery Mode (JMS Only) on page 45 for details.
Durable Name	(Subscribers for JMS Topic Sessions only) Name of the durable Subscriber.

Server Endpoints

Servers are used by an adapter service that communicates with a remote or local client (request-response server).

TIBCO Rendezvous Server Endpoint Reference

You can add a Server endpoint for a TIBCO Rendezvous Reliable, Certified, or Distributed Queue session.

Table 21 TIBCO Rendezvous Server Endpoint Reference

Name	Description
Name	Name of this Server.
Description	Optional description of the Server.
Endpoint Type	Type for the current Server. • For a TIBCO Rendezvous Reliable Subscriber, the default value is Rv RPC Server. • For a TIBCO Rendezvous Certified Subscriber, the default value is RvCm RPC Server. • For a TIBCO Rendezvous Distributed Queue Subscriber, the default value is RvCmq RPC Server.

Table 21 TIBCO Rendezvous Server Endpoint Reference (Cont'd)

Name	Description
Startup State	State when starting up the endpoint. Available options are: • Active (Default) • Inactive
Subject	Subject with which this Server communicates with Clients. The default value is Subscriber.

JMS Server Endpoint Reference

You can add a Server endpoint for a JMS Topic or Queue session.

Table 22 TIBCO JMS Server Endpoint Reference

Name	Description
Name	Name of this Server.
Description	Optional description of the Server.
Endpoint Type	Type for the current Server.
Destination	Destination with which this Subscriber will receive messages.
Message Selector	A message selector is a string that lets a client program specify a set of messages, based on the values of message headers and properties. A selector matches a message if, after substituting header and property values from the message into the selector string, the string evaluates to `true`. Consumers can request that the server deliver only those messages that match a selector.
Delivery Mode	(Subscribers for JMS Topic Sessions only) Delivery mode for the messages. Available options are: • Durable (Default) • Non-Durable See Guideline for Configuring the Delivery Mode (JMS Only) on page 45 for details.

Table 22 TIBCO JMS Server Endpoint Reference (Cont'd)

Name	Description
Durable Name	(Subscribers for JMS Topic Sessions only) Name of the durable Server. The default value is the service name.

Configuring Logging Options

The adapter uses log sinks to generate logs. The adapter defines traces with different roles and sends them to log sinks with the corresponding role. You can either use the default standard I/O for logging or fine-tune where and when different types of information are sent by defining sinks and mapping each sink to one or more roles. Use the Logging tab to configure the logging options. The information is grouped in panels.

- If you use the console window for logging, select the **Log To Standard I/O** check box in the Logging panel in this tab. You can send the information to multiple locations, and you can choose to log one or more message types. See Configuring Logging through Standard I/O on page 60.
- If you are using custom roles, use the All Log Sinks panel to add or remove log sinks and their roles. Then configure the logging options for selected sinks in the Configuration panel. See Adding Log Sinks and Roles on page 62 and Configuring Log Sinks on page 62.

See Appendix E, Trace Messages, on page 275 for trace messages that the adapter can log to a log sink.

Configuring Logging through Standard I/O

To configure the logging through standard I/O, use the Logging panel.

The following table lists the options in the Logging panel.

Table 23 Standard I/O Logging Options

Field	Description
Log to Standard I/O	Use this check box to turn on or off the sending of logging information to the console window when the adapter is started: • Selected (Default) • Cleared Logging information is displayed when this check box is selected.

Table 23 Standard I/O Logging Options (Cont'd)

Field	Description
Log File	File to which logging information is written. The default log file name for the adapter is `%%DirTrace%%/%%Deployment%%.%%InstanceId%%.log`. The log is written to the logs directory specified by the `tibco.clientVar.DirTrace` property in the adapter properties file. If no file name is specified, trace information is not written to a file. You can use module properties to specify the location of the log file. See Configuring Module Properties on page 12 for details on how to define and use module properties.
Log Info Messages	Use this check box to turn on or off the sending of all messages of type `INFO` to the specified location(s): • Selected (Default) • Cleared For when to turn on the option, see Supported Log Roles on page 62.
Log Debug Messages	Use this check box to turn on or off the sending of all messages of type `DEBUG` to the specified location(s): • Selected • Cleared (Default) For when to turn on the option, see Supported Log Roles on page 62.
Log Warning Messages	Use this check box to turn on or off the sending of all messages of type `WARNING` to the specified location(s): • Selected (Default) • Cleared For when to turn on the option, see Supported Log Roles on page 62.
Log Error Messages	Use this check box to turn on or off the sending of all messages or type `ERROR` to the specified location(s): • Selected (Default) • Cleared For when to turn on the option, see Supported Log Roles on page 62.

Adding Log Sinks and Roles

Use the All Log Sinks panel to add or remove a log sink and its roles.

Supported Log Sinks

The adapter supports the following log sinks at runtime:

- **File sink** A file sink sends messages to a file.
- **Standard I/O sink** A stdio sink sends messages to standard input and output.
- **Network sink** A network sink sends messages over the network.
- **Hawk sink** A Hawk sink sends messages to TIBCO Hawk.

Supported Log Roles

For a selected log sink, you can add to it one to many log roles that decide the log levels of the sink.

The log roles available are:

- Debug
- Error
- Information
- Warning

The logging messages generated depend on the roles you add to the log sink.

The use of levels can affect the performance of the adapter. It is recommended that you turn on the required levels only. By default, the Info, Warning, and Error levels are selected. The Debug level is reserved. Do not select this level unless requested by the TIBCO Product Support Group. This option writes a great deal of debugging information to the log file and significantly reduces the speed of the adapter.

Configuring Log Sinks

Select the log sink from the All Log Sinks panel, the configuration options for the sink are displayed. The log sink you select determines the options you need to configure.

- File Sink Configuration Reference, page 63
- Standard I/O Sink Configuration Reference, page 63
- Network Sink Configuration Reference, page 64

- Hawk Sink Configuration Reference, page 64

File Sink Configuration Reference

The following table lists the options for file sinks.

Table 24 File Sink - Configuration Options

Name	Description
Name	Name of the sink.
Description	(Optional) Description of the sink.
File Name	Module property that includes the path and name of the trace file. Extension `.log` is recommended.
File Limit (bytes)	Maximum size of the file, in bytes. Default is 30000. Maximum is 2147483647 bytes.
File Count	Number of rollover files. Default is 3.
Append Mode	Use this check box to control whether to add traces to the existing file at startup: • **Selected** Traces are added to the existing file at startup. • **Cleared** The existing file is overwritten at startup if one of the same name exists.

Standard I/O Sink Configuration Reference

The following table lists the options for standard I/O sinks.

Table 25 Standard I/O Sink - Configuration Options

Name	Description
Name	Name of the sink.
Description	(Optional) Description of the sink.
Output Stream	Output information logged: • stdout: general output information. • stderr: error information.

Network Sink Configuration Reference

The following table lists the options for network sinks.

Table 26 Network Sink - Configuration Options

Name	Description
Name	Name of the sink.
Description	(Optional) Description of the sink.
Subject	Subject of TIBCO Rendezvous messages to be sent.
Session Reference	Click **Browse** and select one of the sessions you define.

Hawk Sink Configuration Reference

The following table lists the options for Hawk sinks.

Table 27 Hawk Sink - Configuration Options

Name	Description
Name	Name of the sink.
Description	(Optional) Description of the sink.
MicroAgent Name	Name of the microagent for traces from this Hawk sink.

Configuring Monitoring Options

TIBCO Hawk monitors the runtime adapter. You can configure the monitoring options using the Monitoring tab.

The following table lists the monitoring options.

Table 28 Monitoring Options

Name	Description
Enable Standard MicroAgent	(Optional) Use this check box to turn on or off the standard TIBCO Hawk microagent. Selected by default.
Standard MicroAgent Name	(Optional) Name for the standard microagent that will be registered with the TIBCO Hawk system. In most cases, keep the default value. You do not need to specify the `InstanceId` variable, because it is automatically set at runtime by the runtime adapter.
Standard MicroAgent Timeout (ms)	(Optional) Timeout value for the standard microagent in milliseconds. The default value is `10000`. Normally you do not need to change this value. However, on machines under extreme stress where method invocations are timing out, you can use this option to increase the timeout value.
Enable Class MicroAgent	(Optional) Use this check box to turn on or off the instance-specific or class-specific standard TIBCO Hawk microagent.
Class MicroAgent Name	(Optional) Name for the class microagent that will be registered with the TIBCO Hawk system. In most cases, keep the default value. You do not need to specify the `InstanceId` variable, because it is automatically set at runtime by the runtime adapter.

Table 28 Monitoring Options (Cont'd)

Name	Description
Class MicroAgent Timeout (ms)	(Optional) Timeout value for the class microagent in milliseconds. The default value is `10000`. Normally you do not need to change this value. However, on machines under extreme stress where method invocations are timing out, you can use this option to increase the timeout value.
Default Microagent Session	(Optional) TIBCO Rendezvous session to be used by the TIBCO Hawk microagents by default. This field is disabled and you cannot change it. The session name and the corresponding session are automatically generated. However, you can modify the session parameters if required by using the Transport tab and modifying the session parameters.

Configuring Advanced Options

The following table lists the advanced options.

Table 29 Advanced Options (Sheet 1 of 4)

Name	Description
Reconnection Configuration	
Maximum Number of Reconnect Attempts	The total number of reconnection attempts to make after the service has been suspended. When this number is reached, the runtime adapter or adapter service will be stopped. For example: A value of `3` means reconnection attempts will continue 3 times A value of `-1` means reconnection attempts will continue indefinitely. **Note:** • If you set the value for parameter `adb.RetryTotal <total reconnection attempts>` in the `adbagent.tra` file, then the value you set in the Designer will be replaced. • To avoid losing data, TIBCO recommends that you need to configure the preregistered listeners for a TIBCO Rendezvous Certified transport session.
Number of Reconnect Attempts Before Suspending Impacted Service(s)	Specify the number of reconnection attempts to make before suspending the service. This value is `1` and cannot be changed.
Interval between Reconnect Attempts (milliseconds)	Specify the time interval (in milliseconds) to elapse between each reconnection attempt. **Note:** If you set the value for parameter `adb.SleepBetweenRetries <milliseconds of sleep between two reconnection attempts>` in the `adbagent.tra` file, then the value you set in TIBCO Business Studio will be replaced.
Adapter Services General Configuration	

Table 29 Advanced Options (Sheet 2 of 4)

Name	Description
Use Exception Table	Select this check box to use an exception table. The exception table is defined when you create a Subscription Service.
Number of Request-Response Service Default Session Threads	Indicates the number of request-response threads to use. Valid values are from 1 through n. Each thread has a separate connection to the database. Specifying multiple threads allows you to load balance incoming RPC requests. **Note:** If you set the value for Number of Request-Response Service Threads at the adapter service level, the service level configuration takes higher precedence over the adapter level configuration.
General Information	

Termination Subject or Topic	As long as a message is sent on the termination subject (if TIBCO Rendezvous is the transport) or topic (if JMS is the transport) then the adapter will stop. The default value of the termination subject or topic is given next.
	`%%Domain%%.%%Deployment%%.adb.%%InstanceId%%.exit`
	The termination subject for a 4.0 project is shown below. Do not change it.
	`_ADB.%%InstanceId%%.TERMINATE`
	See *TIBCO Rendezvous Concepts* for information about specifying subject names. See *TIBCO Enterprise Message Service User's Guide* for information about publishing on a topic.

Table 29 Advanced Options (Sheet 3 of 4)

Name	Description
JMS Destination Prefix	Specify this prefix if you are using the JMS transport type and have selected the Use Separate Service Thread option for a Subscription Service. The destination prefix is the subject used by the single-threaded Subscription Service. By default, the adapter uses a dynamic destination that is generated using the Domain and Deployment module properties, and the adapter configuration name. If you use this default dynamic destination, make sure the values for Domain and Deployment are not empty. You can override the default dynamic destination by specifying the static destination in this field. The static destination must be defined with write permissions on the TIBCO Enterprise Message Service server before it can be used by the runtime adapter.
Debug Level	This field is valid only when the Debug logging option is selected, as described in Configuring Logging Options on page 60. Select how much debugging output to provide at the console window or log file location specified in the Log File field. The options are: • Log no debug information • Log SQL commands executed against the database • Log binding data for each SQL command • Log all debug information
Generate Verbose Output	This field is valid only when the Information logging option is selected, as described in Configuring Logging Options on page 60. Select this check box to include verbose output (all available information) at the console window or log file location specified in the Log File field.
Generate Payload on Error	Select this check box to log the Subscription Service reply payload information under Error role when an error occurs in the Subscription Service.

Table 29 Advanced Options (Sheet 4 of 4)

Name	Description
Script File	Allows you to change the location where the SQL script file is written.
Alerter Name	The name of the alerter service register in the database.

Validating an Adapter Configuration

As you work with resources in TIBCO Business Studio, you can choose to run various builders automatically or manually to log problems, errors, or warnings in the Problems view. The build options are available from the Project menu in the main menu:

- To run the builders automatically, click **Project > Build Automatically**.
- To build one project, make sure **Build Automatically** is unchecked from the **Project** menu. Then select the project in the Project Explorer view, and click **Project > Build Project**.
- To build all projects in the workspace, make sure **Build Automatically** is unchecked from the **Project** menu. Then click **Project > Build All**.

When you save an adapter configuration that contains syntax errors, these errors will be logged in the Problems view. Fix the errors before you test the adapter configuration.

Testing an Adapter Configuration

The Adapter Launcher is a tool for testing the adapter configuration.

Prerequisites

To test the adapter configuration, you must have the runtime component of the adapter installed on your machine.

It is highly recommended that you validate the adapter configuration before running the test. See Validating an Adapter Configuration on page 71.

To Test an Adapter Configuration

Use the Adapter Launcher to test an adapter configuration.

You can run the Adapter Launcher in one of the following ways:

- In the Project Explorer view, right-click the `.adadbmodel` file. From the pop-up menu, click **Run as > Adapter Launcher**.

  Before using this approach, you need to specify the **Working Directory** field of Adapter Launcher in adapter preferences. See Configuring Adapter Preferences on page 11 for details.

- Open the Adapter Launcher and configure the test details:

  a. Open the Adapter Launcher in one of the following ways:

     - From the main menu, click **Run > Run Configurations**.

     - From the toolbar, click the down arrow of the ▶ button and click **Run Configurations**.

  b. In the Run Configurations dialog, right-click **Adapter Launcher**, and click **New**.

  c. In the displayed Configuration panel on the right, provide the following information, and then click **Apply**:

Table 30 Adapter Launcher Options

Name	Description
Name	Name of the adapter launcher for each adapter configuration.
Adapter Configuration	Click **Browse** to select the adapter configuration you want to test. All adapter configurations in this workspace are displayed.

Table 30 Adapter Launcher Options (Cont'd)

Name	Description
Adapter Executable	Choose the adapter executable from the list. Make sure that the version of the runtime matches the version of the configuration. The adapter executable to run your packaged adapter will be shown from a list of choices (each matching a particular installation).
Working Directory	Click **Browse** to supply the directory. The Adapter Launcher creates the necessary runtime and support files required by the adapter in this directory. It is recommended that you do not edit the files in the working directory. Ensure that the disk where the working directory is located contains enough space to save multiple copies of your project.

d. Click **Run**.

When the adapter configuration is running, the display changes to the Console view. A 🟥 button highlights the running adapter. You can now observe the adapter as it is running.

Click the button when you wish to stop the adapter.

When you set up an adapter configuration in the Adapter Launcher, the configuration is saved for each adapter configuration in the project. However, you need to change some of the settings if you want to run the adapter configuration through the Adapter Launcher on a different machine as some of the directories will not be valid.

Chapter 4 Working with Tables and Stored Procedures

Database schema objects, including tables or stored procedures, describe or constrain data or operations used by the adapter. For Publication Services and Subscription Services, the adapter uses tables from the database for getting source data or updating target data. For Request-Response Services, the adapter uses SQL statements or stored procedures to process requests and return results.

You can use tables or stored procedures from different data sources in the same project.

Topics

Overview of Table and Stored Procedure Management

TIBCO Adapters use schema data to describe or constrain data or operations used by the adapter. In TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio), the schema data refers to tables or stored procedures fetched from the database. For inbound data, the fetched objects are used as a constraining mechanism, that is, the incoming data must conform to a structure. For outbound data, the adapter uses fetched objects to interpret data from the application. Fetched tables or stored procedures effectively isolate the data description from the application data itself.

By using the Schema Browser view provided with TIBCO ActiveMatrix Adapter Framework, you can add and manage schemas.

Working with tables and stored procedures includes the following tasks:

1. Add a destination.

   You can add destinations for multiple back-end data sources of different types, for example, SAP, Siebel, or databases. Each destination represents the root of the repository, and when expanded, display a hierarchical structure representing the organization of the back-end system. If it is a destination for a database connection, when expanded, nodes for tables and stored procedures/functions will be displayed.

   See Adding a Destination on page 78.

2. Fetch tables or stored procedures.

   You can fetch tables or stored procedures from the added destinations into the workspace. The fetched objects can then be used by all resources in the project.

   — For Publication Services and Subscription Services, you need to fetch tables.

   — For Request-Response Services, you need to fetch stored procedures or functions.

   See Fetching Tables or Stored Procedures on page 80.

Difference between Database Schema and TIBCO ActiveEnterprise Schema

When you create an adapter service in an adapter configuration with a chosen table or stored procedure, the adapter user interface creates a folder in the `AESchemas/ae/ADB` folder for the adapter configuration and transforms into *AE Schema*. Different from database schema objects, AE Schema are TIBCO proprietary Active Enterprise Schema recognized by TIBCO Adapter SDK, the fundamental class library used in the adapter implementation.

The content of the AE Schema is made up of five parts: associations, classes, scalars, sequences, and unions. The transformation is done automatically, and it is not recommended to modify the AE Schema manually, unless in few cases of the adapter configuration.

Adding a Destination

You can add a database destination to the Schema Browser view by launching the Application Explorer Wizard.

To launch the wizard and add a destination:

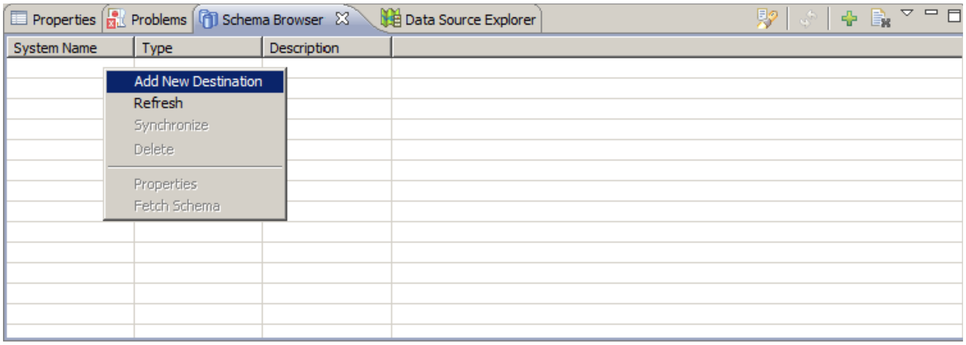
1. Open the Schema Browser view in one of the following ways:

   — From the main menu, click **Window > Show View > Other** to open the Show View dialog. Then click **TIBCO Adapters > Schema Browser** and click **OK**.

   — Click the **Browsing Schema** link in the Adapter Services tab of the adapter configuration editor.

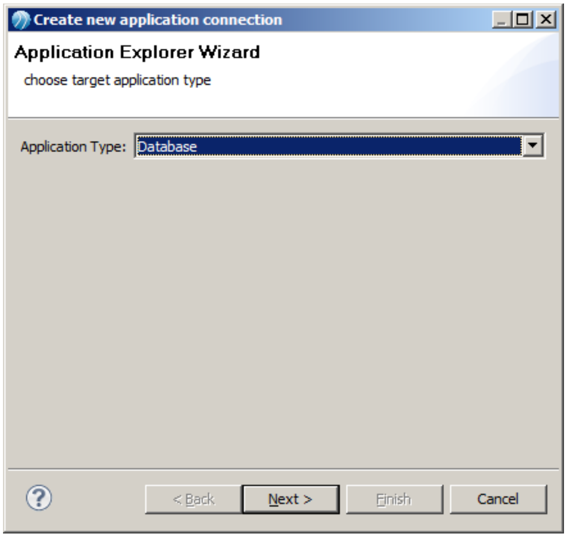
2. Launch the Application Explorer Wizard in one of the following ways:

   — Right-click the Schema Browser, and click **Add New Destination** from the pop-up menu.



   — Click the **Add** button from the action bar of the Schema Browser.

3. In the displayed Application Explorer Wizard, select **Database** as Application Type. Click **Next**.



Create new application connection

Application Explorer Wizard

choose target application type

Application Type: Database

4. In the Basic Information page, specify **Display Name** and provide a description if necessary. Click **Next**.

5. Specify the database connection details and click **Test Connection** to make sure the information is correct.

   See Configuring the Database Connection on page 37 for details.

6. Click **Finish**.

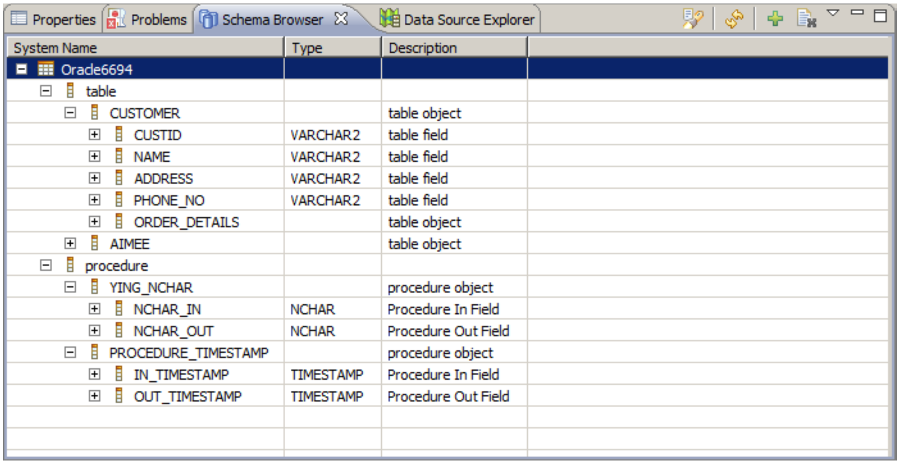
When you finish this procedure, a destination with the name you specified is added in the Schema Browser and you can expand it to show top-level objects.

For the adapter to work correctly, the database connection in the schema browser has to be the same as the one you configured in the adapter configuration.

Fetching Tables or Stored Procedures

You can fetch two types of schema objects from a database destination: tables or stored procedures/functions. Use the Schema Browser view to fetch these objects.

The following figure shows the Schema Browser with fetched tables and stored procedures.



System Name	Type	Description
Oracle6694		
table		
CUSTOMER		table object
CUSTID	VARCHAR2	table field
NAME	VARCHAR2	table field
ADDRESS	VARCHAR2	table field
PHONE_NO	VARCHAR2	table field
ORDER_DETAILS		table object
AIMEE		table object
procedure		
YING_NCHAR		procedure object
NCHAR_IN	NCHAR	Procedure In Field
NCHAR_OUT	NCHAR	Procedure Out Field
PROCEDURE_TIMESTAMP		procedure object
IN_TIMESTAMP	TIMESTAMP	Procedure In Field
OUT_TIMESTAMP	TIMESTAMP	Procedure Out Field

The fetch action following a query only downloads the descriptors for various fetched objects and not the entire object content. The descriptor contains information, such as name of the fetched object, associated connection, columns or parameters of fetched object, data type of columns or parameters, and so on. Actual object is downloaded when it is really required like during service creation or schema import.

Fetching Tables

Each Publication Service or Subscription Service must be associated with a table that describes the data the service receives from or sends to the database.

You can use either the Schema Browser view or the Schema tab of an adapter service to fetch tables and child tables from a database destination. This section introduces how to fetch tables using the Schema Browser. For how to fetch tables using the Schema tab, see the corresponding chapter for each adapter service.

To Fetch a Table

To fetch a table, follow these steps:

1. In the Schema Browser, expand the database destination for which you want to fetch a table.
2. Right-click the **table** node and from the pop-up menu, click **Fetch Table**.
3. In the displayed Table Name Pattern dialog, enter the search criteria. Click **OK**. For example, if you want to fetch a table that starts with "`CUSTOMER`", you can specify "**`CUSTOMER%`**" as the search criteria.

   For how to specify an effective search criteria, see Guidelines for Specifying the Search Criteria on page 83.
4. In the Table Download dialog, select the table you want to fetch and click **OK**.

   If you want to reference an external schema, you can specify the schema name in the **Other Schema** field and click **Add From Other Schema** to load its tables. However, the user must have the proper access privileges to the referred schema. See Referencing an External Schema on page 83 for more information.

To Fetch Child Tables

Data models typically contain tables that share column data through a relationship. You can configure a Publication Service or Subscription Service to include related data from another table as a child table when it publishes or subscribes.

To fetch a child table, follow these steps:

1. In the Schema Browser, expand the database destination for which you want to fetch a table.
2. Expand the **table** node and select a parent table.
3. Right-click the parent table node and from the pop-up menu, click **Fetch Child Table**.
4. In the displayed Table Name Pattern dialog, enter the search criteria. Click **OK**. For example, if you want to fetch a table that starts with "`CUSTOMER`", you can specify "**`ORDER_DETAILS%`**" as the search criteria.

   For how to specify an effective search criteria, see Guidelines for Specifying the Search Criteria on page 83.
5. In the Table Download dialog, select the table you want to fetch and click **OK**.

   If you want to reference an external schema, you can specify the schema name in the **Other Schema** field and click **Add From Other Schema** to load its tables. However, the default schema must have the proper access privileges to the referred schema. See Referencing an External Schema on page 83 for more information.

Repeat this procedure if you want to fetch more child tables.

See Chapter 2, Parent-Child Example in *TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) Examples* for a detailed procedure about fetching parent and child tables.

Fetching Stored Procedures

Each Request-Response Service must be associated with a stored procedure or a SQL statement that describes the client request an adapter needs to execute.

You can use Schema Browser to fetch a stored procedure from a database destination.

To fetch a stored procedure, follow these steps:

1. Expand the database destination in the Schema Browser view.
2. Right-click the **procedure** node and from the pop-up menu, click **Fetch Procedure**.
3. In the displayed Store Procedure Download dialog, select the stored procedure/function you want to fetch and click **OK**.

   You can also specify the following values in the dialog:

Table 31 Store Procedure Options

Field	Description
Name	Enter a unique name for the call operation.
Catalog/Package	(Optional) Only applicable to databases that have more than one catalog or package. The catalog or package in which the procedure resides. This name is used to resolve naming conflicts if more than one catalog or package in the database has the selected procedure with the same name. See your database documentation for more information about catalogs and packages.
Schema	(Optional) The schema is the collection of database objects. Schema objects are the logical structure that directly refer to the database data. In TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio), this is a mandatory field for most databases. Generally, the default schema's name need to be the same as the username.
One Way	(Optional) Select this check box to execute the procedure by using a one-way operation. Clear the check box to execute the procedure by using a two-way operation. If you select the one way operation to execute the procedure, the service will send a message without expecting a response.

TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) retrieves the signature of each stored procedure and function from the database. If you change the stored procedure or database connection while editing your project, you must return to this dialog and click the **Refresh** button to retrieve the changes from the database.

Guidelines for Specifying the Search Criteria

You can choose to fetch a set of tables at one time from the database. But you can only fetch one stored procedure or function for each time, because a unique name must be specified for each stored procedure or function.

Referencing an External Schema

To reference an external schema, the user must have the proper access privileges. These are set in a command line. In the following syntax, *database_username* refers to the user name in `create_user.sql`.

- For Oracle, log in as `system` and grant `create any trigger` and `drop any trigger` permissions to the user. For example:

  ```
  grant create any trigger to database_username
  grant drop any trigger to database_username
  ```

- For Sybase, execute the following command before creating catalog tables for the external schema:

  ```
  sp_role "grant", sa_role, database_username
  ```

In addition, Oracle and Sybase users must have permission to `SELECT` from a source table in an external schema. If table relationships are used, `SELECT` permission is required for both parent and child tables. `SELECT`, `INSERT`, `UPDATE`, and `DELETE` permissions are required for accessing a destination table in an external schema.

- For SQL Server, log in as `sa` and then execute the following command before creating catalog tables for the external schema:

  ```
  use master
  EXEC sp_addsrvrolemember 'database_username', 'sysadmin'
  ```

- For DB2 on IBM i/AS400, you can avoid table access problems by changing the ActiveDatabase user authority to `*ALLOBJ`.

Replicating Schema Information

You can replicate schema information of the whole workspace from another workspace.

Follow these steps to replicate schemas:

1. Open TIBCO Business Studio and create a new project.

   See Creating a Project on page 10 for details.

2. In the project, create an Adapter for Database Configuration.

   See Creating Adapter Configurations on page 33 for details.

3. Open the adapter configuration in the editor, create an Adapter for Database Connection.

   See Configuring the Database Connection on page 37 for details.

4. In the workspace from which you want to replicate the schema information, copy the `.appexplorerrep` file from *source_workspace_directory*`/.metadata/.plugins/com.tibco.adapter.tool.app.explorer`

5. Replace the `.appexplorerrep` file in the current workspace with the copied file.

6. In the current workspace, right-click the Schema Browser view. From the pop-up menu, click **Refresh**.

The schema information will be refreshed as the information from the source workspace.

Chapter 5 Working with BusinessWorks Processes

Processes in TIBCO ActiveMatrix BusinessWorks capture and manage the flow of business information in an enterprise between different data sources and destinations. The Adapter palette installed with TIBCO ActiveMatrix Adapter Framework provides activities for wiring adapter services into processes. You need to map and transform the input data of each activity properly.

See also:

- *TIBCO ActiveMatrix Adapter Reference* for reference information of each adapter activity.

Topics

Overview of Process Development

A process definition is a graphical representation of your business process model. You can develop and test business processes by using TIBCO Business Studio.

Working with processes includes the following tasks:

1. Create a process

If the Project name contains a hyphen, you cannot create a BusinessWorks process under the project.

2. Add activities

   Select the palette that has the activities you want to add to the business process. In particular, the Adapter palette has activities for communicating with adapter services. Configure the selected activities by clicking an activity and specifying the options available as tabs in the Properties view.

   See *TIBCO ActiveMatrix Adapter Reference* for more information about each adapter activity.

3. Create transitions between activities
4. Map and transform input data

   Perform mapping and transformation of data for the input of each activity.

   For the mapping operations for activities, see *TIBCO ActiveMatrix BusinessWorks Bindings and Palette Reference.*

5. Validate a process
6. Test a process

See *TIBCO ActiveMatrix BusinessWorks Application Development* for details on process development.

Workflows with Activities and Processes

TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) and the TIBCO Adapter palette provide flexible ways to create adapter activities for processes.

You can create a new adapter activity from the Adapter palette in one of the following ways:

- With the `.bwp` process file opened in the Process Editor, drag the activity from the Adapter palette in the Palette view to the Process Editor.
- Right-click the opened `.bwp` process file in the Process Editor. From the pop-up menu, click **Add Resource > Adapter Palette >** ***Activity_to_Add***.

Testing a BusinessWorks Process

Similar to testing an adapter configuration, in TIBCO Business Studio, you can test a BusinessWorks process in a few ways:

If only one project is in the workspace, to test the process, follow these steps:

1. From the Project Explorer view, expand ***project_name*** **> Module Descriptors**, and double-click **Components**.
2. In the Components editor, select the processes you do not test, and click the ✖ button on the right. This removes the processes not to test from the project Components.
3. Save the project.
4. In the Project Explorer view, right-click the **`.bwp`** file. From the pop-up menu, click **Run as > Launch BusinessWork**s. Only the processes in the project Components will be executed.

If more than one projects are created and opened in the workspace, follow these steps:

1. Open the Run Configurations dialog in one of the following ways and configure the test details.
   - — From the main menu, click **Run > Run Configurations**.
   - — From the toolbar, click the down arrow of the ▶ button and click **Run Configurations**.
2. In the Run Configurations dialog, expand **BusinessWorks Application > BWApplication**.

3. Click the Applications tab. Select the application you want to test and click **Run**.

   If you want to test certain processes in the application, you need to first remove the other processes from the project Components before you use the Run Configuration dialog for testing. Repeat step 1 to step 3 of the first approach to do this.

Activities for Communicating with Adapter Services

The Adapter palette contains activities for communicating with configured adapter services. The following table lists the adapter services and associated activities in the Adapter palette.

See *TIBCO ActiveMatrix Adapter Reference* for more information on when and how to use these activities.

Table 32 Adapter Services and Associated Activities in the Adapter Palette

Adapter Service	Associated Activities
Publication Service	**Adapter Subscriber** (Process Starter[1]) Starts a process based on the receipt of a message from the Publication Service of the specified adapter.
	Wait for Adapter Message Waits for the receipt of a message from the Publication Service of the specified adapter.
Subscription Service	**Publish to Adapter** Publishes a message received by the Subscription Service of the specified adapter.
Request-Response Service	• Request-Reply Mode **Invoke an Adapter Request-Response Server** Communicates as a client with the Request-Response Service of the specified adapter. • RPC Mode **Publish to Adapter** Publishes a message received by the Subscription Service of the specified adapter.

[1] Timer, File Poller, and any of the process starter activities are activities used to start a process when an event occurs.

Handling the anyType AE Data Type

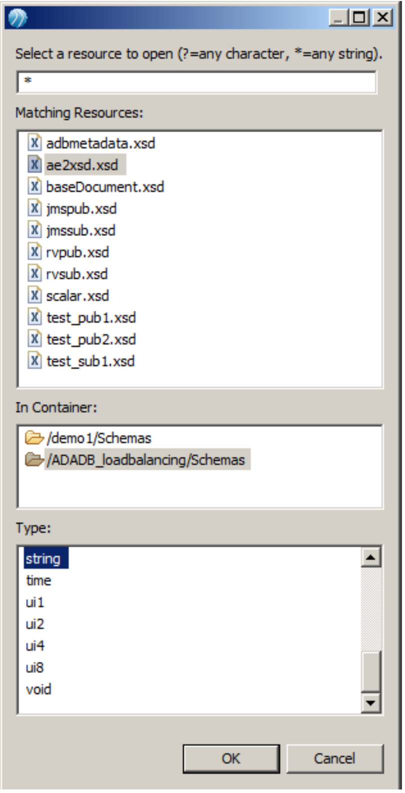
In the Input tab of Adapter activities, you cannot set value for any variable that is of the anyType AE Schema data type. In particular, this data type is used in the CLOSURE and DATA elements of activities that are associated with Request-Response Services.

To set the value of the variable, you need to substitute its type with Simple Type and select the appropriate type that suits your needs from `ae2xsc.xsd`. The detailed procedure is as follows:

1. In the Input tab of the activity, right-click the variable that you want to coerce a new type.
2. From the pop-up menu, click **Substitution**.
3. In the Substitution dialog, specify the following fields:
   - **Component Type**: select **Simple Type** from the list.
   - **Namespace**: click the Browse button to open the dialog for selecting the data type. Select **`ae2xsd.xsd`** from the **Matching Resource** list and select

the AE Schema in the current project, and select the data type from the type list.

The following figure shows the data type dialog.



Select a resource to open (?=any character, *=any string).

*

Matching Resources:

- adbmetadata.xsd
- ae2xsd.xsd
- baseDocument.xsd
- jmspub.xsd
- jmssub.xsd
- rvpub.xsd
- rvsub.xsd
- scalar.xsd
- test_pub1.xsd
- test_pub2.xsd
- test_sub1.xsd

In Container:

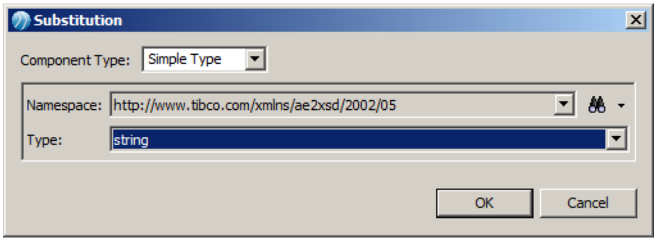
- /demo1/Schemas
- /ADADB_loadbalancing/Schemas

Type:

- string
- time
- ui1
- ui2
- ui4
- ui8
- void

Click **OK**.

The following figure shows the Substitution dialog.



Substitution

Component Type: Simple Type

Namespace: http://www.tibco.com/xmlns/ae2xsd/2002/05

Type: string

Chapter 6 Migrating an Adapter Project Created in TIBCO Designer

You can migrate an adapter project created in TIBCO Designer with TIBCO ActiveMatrix BusinessWorks 5.x to TIBCO ActiveMatrix BusinessWorks 6 by using the migration tool in TIBCO Business Studio.

See also:

- The TIBCO ActiveMatrix BusinessWorks documentation.

Topics

Prerequisites of Migrating a TIBCO Designer Project

The Migration Tool in TIBCO Business Studio only supports migration from TIBCO ActiveMatrix Adapter for Database 7.0 to TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio). Before migrating a TIBCO Designer project, read this section and make preparation accordingly.

Validating the Project in TIBCO Designer

Before you prepare a project for migration, it is critical that you validate it. TIBCO Designer includes reference-checking and other validation facilities that allow you to make sure a project is internally consistent.

See "Validating Projects" in *TIBCO Designer User's Guide* for details.

Notes on Migrating a TIBCO Designer Project

Read the following recommendations and limitations that apply when migrating a TIBCO Designer project:

- You can migrate a TIBCO ActiveMatrix BusinessWorks 5.x project to a TIBCO ActiveMatrix BusinessWorks 6 project, but not in reverse.
- TIBCO Business Studio does not support importing `.dat` files from TIBCO Designer. To migrate a project stored in a `.dat` file, you need to import the file in TIBCO Designer and save it as a multi-file project.
- The name of a project migrated to TIBCO Business Studio cannot contain any of the following characters:

  ```
  (space) ! $ % & + . / @  \ ~
  ```

Migrating a TIBCO Designer Project

To migrate a TIBCO Designer project to TIBCO ActiveMatrix BusinessWorks 6, follow these steps:

1. In TIBCO Business Studio, open the migration tool using one of the follow ways:

   — Click **File > Import**. In the opened dialog, expand the **Migrate BW Projects** node, and select **Migrate BW Projects...**. Click **Next**.

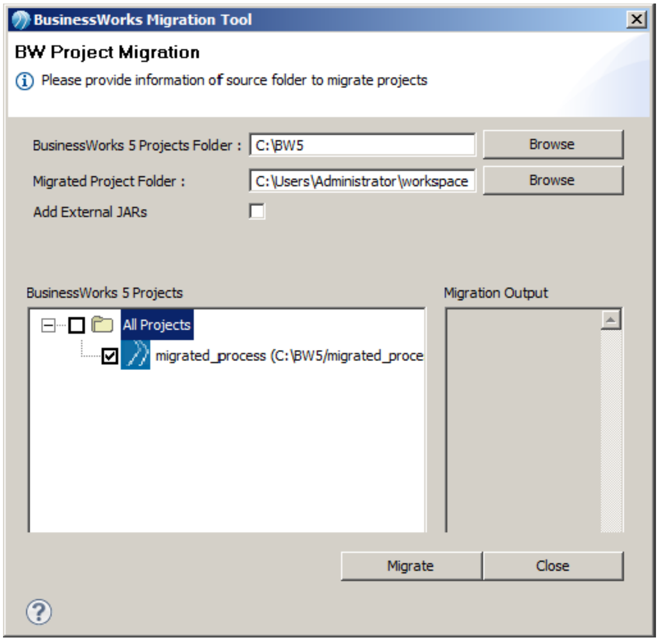
   — Click **Project > Migrate BW Projects**.

2. In the BusinessWorks Migration Tool dialog, specify the following fields:

   — **BusinessWorks 5 Projects Folder**: the source location of the projects to migrate.

   — **Migrated Project Folder**: the target location of the migrated projects.

   — (Optional) **Add External JARs**: Not used in TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio).



BusinessWorks Migration Tool

BW Project Migration

Please provide information of source folder to migrate projects

BusinessWorks 5 Projects Folder : C:\BW5

Migrated Project Folder : C:\Users\Administrator\workspace

Add External JARs □

BusinessWorks 5 Projects

□ All Projects

☑ migrated_process (C:\BW5/migrated_proce

Migration Output

3. After you specify the BusinessWorks 5 project folder, in the displayed project tree, select the adapter projects you want to migrate.

4. Click **Migrate**.

5. When the migration completes, click **Close**.

The migrated projects are displayed in the Project Explorer view.

Post-Migration Tasks

Depending on the configuration of the project to migrate, you need to complete some of the following tasks after migration.

Configuring the Design-Time Connection

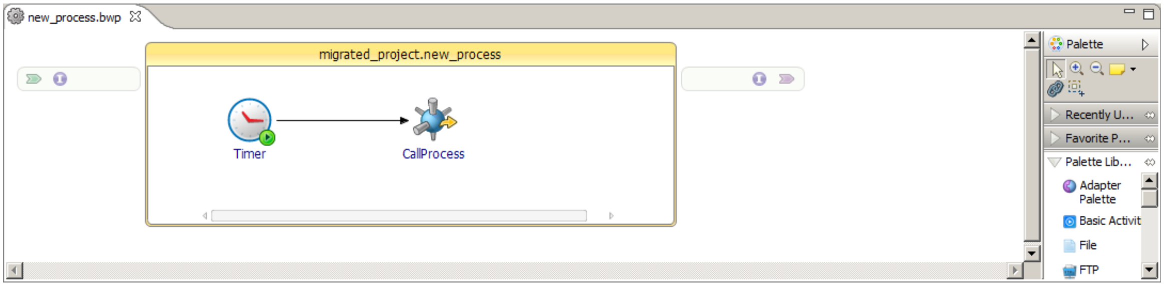
In TIBCO Designer, you can configure a TIBCO ActiveMatrix Adapter for Database to use different connection information at design time and runtime. However, for configuration in this kind, the design-time information will be lost after migration. The connection reference in the migrated project uses the runtime information specified in TIBCO Designer. In order to use a different connection for the design purpose, you need to create an database connection resource and associate the resource with the adapter. See Configuring the Database Connection on page 37 for configuration details.

Calling a Process that Uses the Start Activity

If the migrated project has a process that uses the Start activity, you need to manually create a new BusinessWorks process to call the migrated process:

1. Create a new process in TIBCO Business Studio.
2. In the new process, add a Timer, File Poller or any activity of the process starter type.
3. In the new process, add a CAll Process activity.
4. Add a transition between the process starter and the Call Process activity.

   The following figure shows a sample new process.



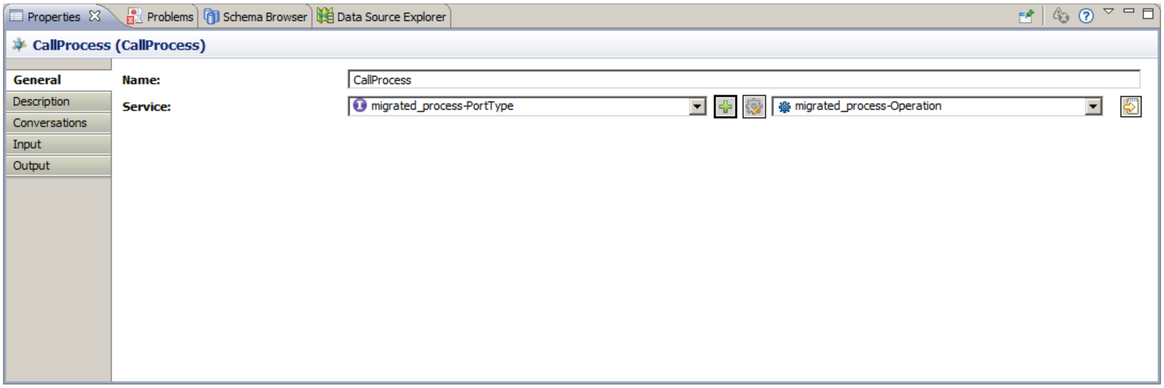
5. Select the Call Process activity and configure the activity:

   a. In the Properties view, click the General tab.

   b. Click the ✚ button next to the **Service** field.

   c. In the opened Select a Service dialog, select the migrated process. Click **OK**.

   The following figure shows the configuration of a sample Call Process activity.

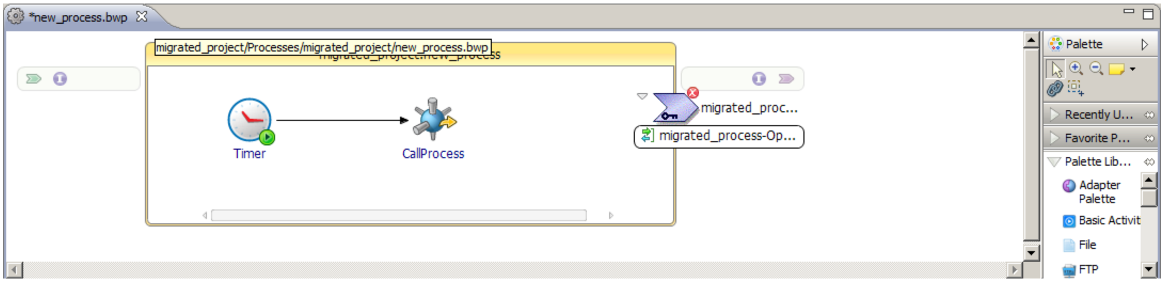


Name: CallProcess

Service: migrated_process-PortType | migrated_process-Operation

6. After you select the process to call, the process reference is displayed in the created process.

   The following figure shows the created process with the reference displayed.



7. Select the reference added to the new process and configure the reference:

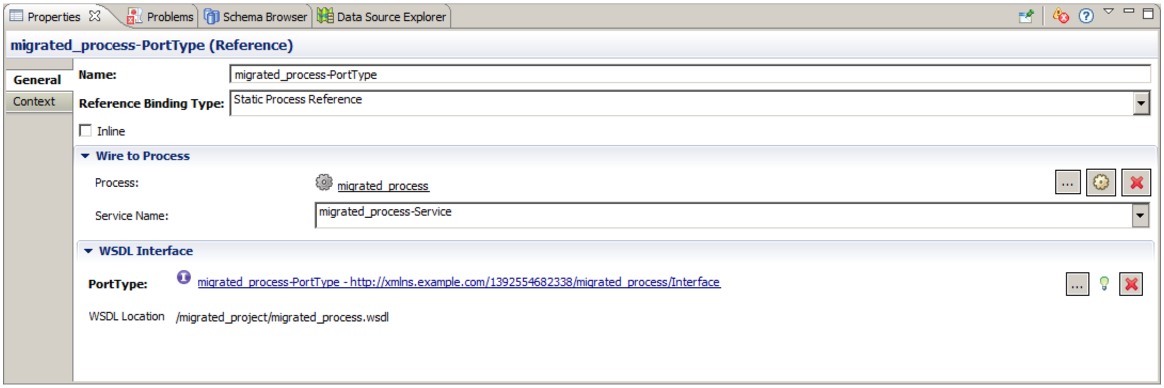
   a. In the Properties view, click the General tab.

   b. Click the **Select Process (...)** button next to the **Process** field under **Wire to Process**.

   c. In the opened Select a BusinessWorks Process dialog, select the migrated process. Click **OK**.

   d. In the Service Name field, select the service from the list.

The following figure shows the configuration of a sample reference.



migrated_process-PortType (Reference)

Name: migrated_process-PortType

Reference Binding Type: Static Process Reference

□ Inline

Wire to Process

Process: migrated_process

Service Name: migrated_process-Service

WSDL Interface

PortType: migrated_process-PortType - http://xmlns.example.com/1392554682338/migrated_process/Interface

WSDL Location /migrated_project/migrated_process.wsdl

8. Save the project.

After you finish these steps, call the created process to invoke the migrated process.

Restoring the Maximum Rows Value in the Customer RPC Operation Level

In TIBCO Designer, you can specify the maximum rows value for a Request-Response Service in the operation level. But after migration, this value set in TIBCO Designer will be lost.

To restore this value, follow the procedure introduced in Maximum Rows on page 162.

Using Internationalized Data

To deal with internationalized data, you need to add parameter `-Dfile.encoding=UTF-8` in the `TIBCOBusinessStudio.ini` file before processing UTF-8 messages.

Chapter 7 Configuring Publication Services

A running Publication Service automatically publishes data when rows in pre-specified database tables are inserted, updated, or deleted. In addition to its name, quality of service, and wire format, you can specify to use different polling methods and storage modes for the adapter service. The adapter uses a publishing table created in the database to store publishing status and detect new rows.

This chapter introduces the specific configuration for Publication Services. For its general configuration, see Configuring the General Information of Adapter Services on page 41. For how to create an adapter service, see Creating Adapter Services on page 33.

Topics

Publication Service Reference

Depending on the database you use, the Publication Service options can be different.

Configuration Tab

Use this tab to configure the general information of the adapter service.

See Configuring the General Information of Adapter Services on page 41.

Schema Tab

Use this tab to specify the schema information of the adapter service. For how to specify the tables, see Specifying the Schema on page 109.

Tool Buttons

You can fetch and configure the tables by using the tool buttons in the Schema tab. The tables added are displayed in the Table panel.

- To add the root table: Click the **Add table** button. This table is the source table to publish from when data is inserted into it.
- To add a secondary table to the source table: Click the **Add child table** button.
- To refresh stored table schema information by retrieving new information from the database, click the **Re-find schema from database** button.
- To load a database table schema, and convert it into an AE schema: click the **Load table schema from database** button.
- To delete a table, right-click the table and from the pop-up menu, click **Delete Table**. If the table has child tables, its child tables are also removed.

Schema Table

The following table lists and explains the table view of AE Schema.

Table 33 Publication Service: Schema Tab—Schema Table View

Field	Description
Allow Key Columns Only	Use this check box to determines whether child columns can be joined to a column that is not key. • **Selected** Child columns are joined only to a key column. • **Cleared** Child columns can be joined to any column.
Tables and Columns	Loaded tables and columns.
Join To	Name of a parent table column to join to for parent-child relationships.
User Key	Click to define the column as a user key.
Update Trigger	Fire a trigger when an UPDATE statement changes a value in the column. **Note:** Only the following databases are supported in the Update Trigger option: • Oracle • SQL Server • DB2 UDB/AS400/OS390 • Sybase
AE Type	Primitive type mapped to an TIBCO ActiveEnterprise type.

Publication Options Tab

When you add a new publishing table, a new entry is inserted into the repository for the publisher adapter. The entry includes the name of a publishing table along with a sequence, stored procedure, and trigger. A class object for the publication is created in the project repository.

Table 34 Publication Service: Publication Options Tab (Sheet 1 of 4)

Field	Description
Publication Service Base Information	

Table 34 Publication Service: Publication Options Tab (Sheet 2 of 4)

Field	Description
Polling Method	The Publication Service uses the following two methods to monitor changes to a database table. • Timer (default setting) It is used to pick up the records from the database in a specified time. This specified time can be set in the Polling Interval field. • Alerter (Only available when using Oracle and Microsoft SQL Server databases) It is used to asynchronously notify running Publication Service instances of database changes. Use the alerter only when database changes are infrequent.
Polling Interval	Type a specific polling interval in milliseconds. This is how often an adapter with a Publication Service checks the publishing table for new rows. The default value is `5000`, or once every five seconds. **Note:** • If you specify a polling interval to `0`, it is assumed that you are using an alerter to manage polling. • If you select the Alerter from the Polling Method drop-down list, the Polling Interval field will be disappeared.
Use Polling Batch Size	Check this check box to determine whether you want to poll specified number of rows during each polling interval or not.
Polling Batch Size (Maximum Rows)	Only active when Use Polling Batch Size check box is selected. The maximum number of messages that are picked up per polling interval. The default value is set to 0. **Note:** When you use Polling Batch Size and Enable Group Messaging together in the Publication Service, it is recommended that you set the value of Polling Batch Size equal or larger than the value of Group Size to increase the performance. If the Polling Batch Size is set smaller than the Group Size, the group cannot be fully filled.
Batch Publish Status Updates	Only active when Use Polling Batch Size check box is selected. Select this check box to optimize publishing performance by batching message status updates to the publishing table. If an adapter stops before a batch update is performed, the status column is not updated. As a result, duplicate messages may be published when the adapter is restarted. Do not use this option when messages are published using a parameterized subject name.

Table 34 Publication Service: Publication Options Tab (Sheet 3 of 4)

Field	Description
Publisher Batch Confirm Size	Applies only to adapters with a Publication Service that use TIBCO Rendezvous certified message delivery. Do not use this option when messages are published using a parameterized subject name. The number indicates how many message status need to be updated in a single batch. Entering a value in this field optimizes performance. However, if an adapter stops before a batch update is performed, the status column is not updated. As a result, messages that were successfully published might still have a status of P (pending) in the publishing table when the adapter is restarted. In this case, the ledger file contains the correct status information. Smaller values in this field decrease this risk.
Publisher Batch Confirm Timeout	Applies only to publisher adapters with publications that use TIBCO Rendezvous certified message delivery. Do not use this option when messages are published using a parameterized subject name. The number of milliseconds to wait before updating the status column. After this interval, an update is performed even if the batch size value is not reached. The default value is `10000` (10 seconds). A value of `0` means that no timeout interval is used. If an adapter instance stops before a batch update is performed, the status column is not updated. As a result, messages that were successfully published might still have a status of P (pending) in the publishing table when the adapter is restarted. In this case, the ledger file contains the correct status information. Smaller timeout values decrease this risk.
Publication Service Option Information	
Storage Mode	For each Publication Service, you must specify a storage mode: • Publish by Value Copies all specified columns in the source table to the publishing table. See Publish by Value on page 129 for more information. • Publish by Reference Copies only key column values to the publishing table. If no key column is defined in the database, a substitute non-key column must be defined to publish by reference. See Publish by Reference on page 130 for more information.
Publishing Table	Name of the database table used to store a copy of data to be published. The table name can be qualified using the *schema.table_name* format. The publishing table cannot contain any user-created columns where the column name starts with `ADB_`. These characters are reserved for use by the adapter. A common practice is to use the publishing table name prefixed by `P_`. For example, if your source table is `MY_ORDER`, its publishing table should be named `P_MY_ORDER`. A publishing table name must be less than 64 characters.

Table 34 Publication Service: Publication Options Tab (Sheet 4 of 4)

Field	Description
Update Mode	Select the following two methods to update tables: • Update updates a row in the destination table only when the row exists. • Upsert updates a row in the destination table if the row exists. If no such row exists, it performs an insert.
Enable Loop Detection	Select to enable loop detection and prevent an infinite loop from occurring when the publishing and destination table are the same table. Loop detection is disabled for DB2 on z/OS.
Do Not Generate Triggers	Select to prevent the generation of triggers. Although this option is not recommended, it allows you to manually manage the insertion of data into the publishing table.
Enable Group Messaging	Select to use group messaging. Note that Rendezvous message wire format does not apply for group message.
Group Size	Displayed only when the Enable Group Messaging check box is selected. Specify the number of rows to publish in a single message. Module Properties are acceptable.
Enable Load Balancing	Select this check box to use load balancing that enables multiple publisher endpoints to poll and publish changes of the same source (publisher) table. Set batch polling size before selecting this option. **Note:** When setting the load balancing for a Publication Service, the added table, the automatically created publisher table, the mutex name, and the message subject name must be same.
Mutex Name	Displayed only when the **Enable Load Balancing** check box is selected. Specify the name for the mutex table.
Number of Publication Service Threads	These threads are allocated on demand. For example, if the Number of Publication Service Threads to 1, a publication thread and the database connection will be created automatically. With setting this number, each Publication Service can hold specified threads that connect the database separately, and process requests in parallel. See Multithreading in Publication Services on page 186 for more information.
Publish Child Data	This check box is selected by default. When selected, the parent row and all related child rows are published if updating a parent row. Upon receipt of such a message, a Subscription Service updates the parent row, then updates all the child rows with the data that was received in the message. For information on adding related tables for a Publication Service, see Adding Child Tables on page 109. For information on adding related tables for a Subscription Service, see Set the Mapping between Child Tables on page 150.
Polling Commit for DB2	Select this check box to enable the adapter publisher to do a commit after selecting a query for DB2.

Child Table Order By Tab

Use this tab to specify the order of child tables.

See Adding Child Tables on page 109.

DB2/390 Options Tab

The DB2/390 Options tab is displayed, when you select **DB2 OS390** from the **Vendor** list in the Configuration tab of the adapter configuration.

When the DB2 load utility loads rows to the source table, it does not activate the INSERT triggers of the table. The loaded data is not published.

Table 35 Publication Service: DB2/390 Options Tab

Field	Description
Database Name	Name of the database that you want to put your publisher table in.
Table Space Name	Name of the table space where the publisher table is located.
Storage Group (optional)	Designator of the storage group that holds the publisher table indexes.
Buffer Pool (optional)	Name of the buffer pool to be used for indexes.
Index Suffix	Suffix of your choice, up to 13 characters, that the adapter appends to each of the indexes (`IDX1_`, `IDX2_`, and `IDX3_`).
Trigger Suffix	Suffix of your choice, up to 5 characters, that the adapter appends to each of the triggers (`T1`, `T2`, and `T3`).

DB2/AS400 Options Tab

The DB2/AS400 Options tab is displayed, when you select **DB2 AS400** from the **Vendor** list in the Configuration tab of the adapter configuration.

Trigger Option

The new data from the source table is copied to the publishing table by setting the trigger options. You can select from the following options:

- SQL—The trigger program is defined entirely using SQL. SQL trigger can be insert, update, or delete triggers. When copying from the source table to the publishing table, the prompt is not returned until all data is written. Currently only SQL trigger can be used for DB2 AS400.

- Synchronous (not applicable)—The trigger is written in RPG. When copying from the source table to the publishing table, the prompt is not returned until all data is written.
- Asynchronous (not applicable)—When copying from the source table, data is inserted into a data queue, then inserted into the publishing table asynchronously. The prompt is not blocked, so you can continue working while data is inserted into the publishing table.

Sybase Specific Options Tab

The Sybase Specific Options tab is displayed, when you select **Sybase** from the **Vendor** list in the Configuration tab of the adapter configuration.

Select the **Overwrite Triggers?** check box to delete and re-create existing triggers in the publishing table.

Specifying the Schema

Each Publication Service must be associated with a table or a set of tables that describe the data that the service receives from the database. You must specify the table information before configuring other options for the service. The Schema tab of the service configuration is for specifying the tables. See Adding Child Tables on page 109 for the explanation of the schema table displayed in the tab.

Schema information includes tables and their columns. Use one of the following ways to specify the tables and columns:

- Use the tool buttons in the Table panel. See Tool Buttons on page 102 for details.
- Drag the fetched tables from the Schema Browser view to this tab. See Fetching Tables or Stored Procedures on page 80 for details.

A Publication Service can send the data inserted in parent and child tables. See Adding Child Tables on page 109 for how to add related child tables in the service schema.

Source table names can be qualified with a database schema name. To access tables in other schemas, the database user specified in the adapter configuration must have the proper set of permissions granted. See Referencing an External Schema on page 83 for details.

If a primary key is not defined for a table, the update and delete triggers will not be generated for the table. To define a primary key for the table, select the User Key column in a table row. The update and delete triggers will then be defined.

Adding Child Tables

Data models typically contain tables that share column data through a relationship. You can configure a publishing table to include related data from another table when it publishes. When rows are inserted into the publishing table, a message that includes data from the source table and related (child) table is published. Data from the related table is not copied into the publishing table, but is fetched by reference.

Any change to the child tables without a change in the parent table will not be processed. The adapter monitors only the parent table for publishing. The adapter updates the child rows by deleting all the related child rows, then inserting child rows again based on the data in the received message.

Adding child tables requires two separate procedures, one for the publisher adapter and another for the subscriber. First, you add child tables for the source table, then you add child tables for the destination table. When you add a child table, TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) creates a class object in the repository for the child table, and an association object for the relationship.

This section explains the procedure of specifying child tables in Publication Services. For the procedure of specifying child tables in Subscription Services, see Subscription Options Tab on page 143.

Restrictions of Child Tables

The following restrictions apply to parent and child tables for a Publication Service is configured to publish messages to a Subscription Service:

- The child table in the source database and child table in the destination database must have the same columns, but the table names can be different. If the child table associated with the publishing table and the child table associated with the destination table have different names, you must set a mapping between the child tables when configuring the Subscription Service.
- When parent-child relationships are defined, a subscriber adapter must use the same repository as the publisher adapter.
- When working with parent-child table, it is recommended to set ADB_OPCODE in the parent table and child table to the same value. Or only set the value of ADB_OPCODE in the parent table and leave the child table empty.

Procedure of Specifying Child Tables

To specify the child tables to publish, complete the following tasks:

- Task A, Add Child Tables, page 110
- Task B, Specify the Relationship between the Parent and Child Tables, page 111
- Task C, (Optional) Specify the Result Set Sequence for Child Tables, page 112

See *TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) Examples* for how to configure an adapter configuration to publish and subscribe to data that is stored in related parent and child tables.

Task A Add Child Tables

To add child tables in the Schema tab, you can either use the tool buttons in the Schema tab, as explained in the following procedure.

1. Click the name of the parent table and select it, then click the **Add child table** button.
2. In the Table name pattern dialog, specify the search criterion.
3. In the Select table dialog, select the related child table from the list.
4. (Optional) if you want to add tables from other schema, specify the schema name in the **Other Schema** field, and click **Add From Other Schema**, then select the related child table from the list.

   To reference an external schema, the default schema must have the proper access privileges. For instructions on granting access privileges to an external schema, see Referencing an External Schema on page 83
5. If you want to select the columns of the table to publish, click **Next** and select the **Use** check boxes next to the columns.
6. Click **Finish**.
7. Repeat step 1 to step 6 to add more child tables.

Task B Specify the Relationship between the Parent and Child Tables

You need to designate a foreign key column as a key in each child table so that a relationship to the parent table can be defined. You must also specify the relationship between the primary column in the parent table and the foreign key column in the child table.

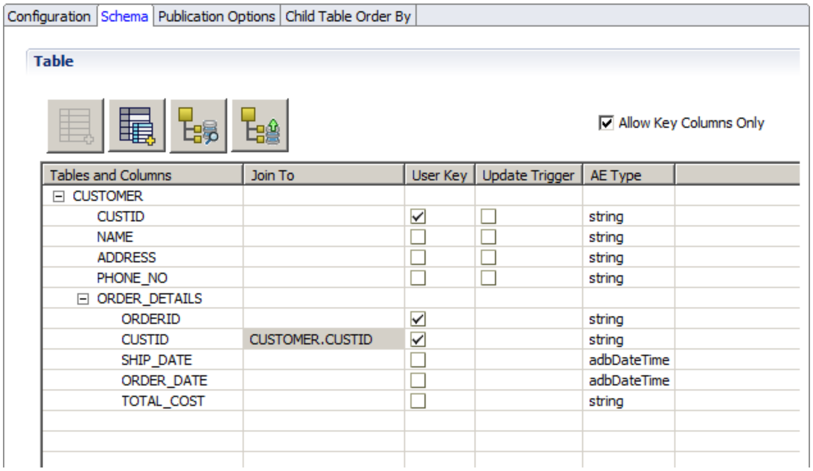
Follow these steps:

1. Select the **User Key** check box for the foreign key column in the child table.
2. Click the **Join To** field next to the appropriate child table column, and select the primary key of the parent table from the list.

   Select the **Allow Key Columns Only** check box, child columns are joined only to a key column. Clear this check box, child columns can be joined to any column.

The following figure shows the relationship specified between the `CUSTOMER` parent table and the `ORDER_DETAILS` child table.

Figure 2 Publication Service: Schema Tab—Add a Child Table



☑ Allow Key Columns Only

Tables and Columns	Join To	User Key	Update Trigger	AE Type
⊟ CUSTOMER				
CUSTID		☑	□	string
NAME		□	□	string
ADDRESS		□	□	string
PHONE_NO		□	□	string
⊟ ORDER_DETAILS				
ORDERID		☑		string
CUSTID	CUSTOMER.CUSTID	☑		string
SHIP_DATE		□		adbDateTime
ORDER_DATE		□		adbDateTime
TOTAL_COST		□		string

Task C (Optional) Specify the Result Set Sequence for Child Tables

You can specify the column(s) of an Order By Clause for each child table query by using the Child Table Order By tab, so that the adapter returns the child table result set in a particular sequence.

To do so, perform the following steps:

1. Click a child table name, then click **Select Columns**.
2. In the Select Child Table Order By Columns dialog, select the column(s) for the Order By Clause from the left panel, then click the **>>** button to move the column(s) into the right panel. Use the **Up** and **Down** buttons to arrange the column order by sequence.
3. If you want to use DESC for a column, select the check box next to the column.
4. Click **Finish**.

See Chapter 2, Parent-Child Example in *TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) Examples* for a detailed procedure about specifying parent and child tables using the Schema tab.

Preparing the Publishing Table

TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) uses publishing tables created in the database to monitor the changes to the source tables and to record the publishing status. You need to specify a publishing table for each table you want to activate.

Publishing Table

Publishing tables mirror tables that you have identified for monitoring. They contain additional columns, primarily a sequence number and delivery status, which are needed by the adapter to detect new rows.

When you specify a publishing table at design time, the adapter creates the publishing table in the database and the trigger that automatically populates the values for these additional columns. It is not recommended that you modify these values.

Specify the Publishing Table

When you add a table to the Schema tab, the adapter does the following for you:

- Specify the publishing table in the **Publishing Table** field located in the Publication Options tab. A common practice is to use the publishing table name prefixed by "`P_`".
- Click the **Class Reference** link in the Schema panel located in the Configuration tab to open the AE Schema. For more details about AE Schema, see Difference between Database Schema and TIBCO ActiveEnterprise Schema on page 76.

When you save the adapter configuration, the adapter creates the publishing table in the database.

If you want to change the name of the publishing table, use the **Publishing Table** field.

Publishing Table Reference

In addition to a copy of the columns of a source table, the publishing table has the following additional columns.

Table 36 Publishing Table: Additional Columns

Column Name	Type	Description
ADB_SUBJECT	VARCHAR2(255)	Used to specify the subject to publish the current row. Length is 255. You can set a message subject in this field which takes precedence over the default service subject. The adapter will publish this row with this new message subject. In group messaging, the group messages are sent to the subject set for the last row of the group.
ADB_SEQUENCE	INTEGER	Stores the monotonically increasing sequence number that represents new rows in the publishing table. If a column with this name exists, the number is generated automatically. By default, the schema type is `string`. The adapter treats this number as a string. This number can be larger than an integer if the database supports it.
ADB_SET_SEQUENCE	INTEGER	Currently not used.
ADB_TIMESTAMP	DATE	Time of row insertion in publishing table that is used to calculate expiration of rows. The timestamp is generated automatically.
ADB_TRACKINGID	VARCHAR2(40)	Tracking ID of the message. This is automatically added to the publishing table and the publishing schema. If you do not want to monitor the tracking id, you can manually remove this field from the project schema and the publishing table

Table 36 Publishing Table: Additional Columns (Cont'd)

Column Name	Type	Description
ADB_OPCODE	INTEGER	Operation code used by an adapter instance: `1` indicates INSERT. `2` indicates UPDATE. `3` indicates DELETE. `4` indicates UPSERT. UPDATE if row exists, otherwise INSERT. `10` indicates BYPASS (See Incremental Parent-child Operation on page 119) If an incoming TIBCO Rendezvous message does not have an operation code, an INSERT occurs. **Note:** When working with parent-child table, it is recommended to set ADB_OPCODE in the parent and child table to the same value. Or only set the value of ADB_OPCODE in the parent table and leave the child table empty.
ADB_UPDATE_ALL	INTEGER	Currently not used.
ADB_REF_OBJECT	VARCHAR2(64)	When publishing by reference object is used, contains the name of the reference object that provides source data.
ADB_L_DELIVERY_STATUS	CHAR	Delivery status of a TIBCO Rendezvous message: `P` indicates pending acknowledgement. `N` indicates that a new message has arrived, but has not yet been published. S indicates that one thread has got this record, and will not be taken again when polling other records. `C` indicates complete. `F` indicates failed.
ADB_L_CMSEQUENCE	NUMBER (38, 0)	Certified messaging sequence number associated with this message.

When publishing a parent-child record or in Publish By Reference mode, if the key value is empty (`null`) in the publishing table, the adapter will consider this as an error operation, and update the failure status in the publication table by setting `adb_l_delivery_status` to `F`. The data will not be published.

The publication table cannot contain any user-created columns where the column name starts with `ADB_`. These characters are reserved for adapter use.

Publication Example

The following is a list of the SQL commands used to create a publication by using Oracle database. The list can be found in the *TIB_ADADB_HOME*`\demo\demo1\demo1_`*database*`.sql` file. The file shows the stored procedure, and insert, update and delete triggers that are generated for a publication.

```
-- *******************************************
--   TIB/Adapter for ActiveDatabase
--     Demo SQL script (Oracle)
-- *******************************************

-- Spool the output of sql commands to a log file spool demo.log

-- *******************************************
-- Create sample tables for RV
-- *******************************************

-- Publication source table
CREATE TABLE ORDER_TABLE (
   ORDER_ID NUMBER PRIMARY KEY,
   ORDER_DESCRIPTION VARCHAR2(128),
   ORDER_PRICE NUMBER(10,3)
);

-- Create the destination table
CREATE TABLE SUB_ORDER (
   ORDER_ID NUMBER PRIMARY KEY,
   ORDER_DESCRIPTION VARCHAR2(128),
   ORDER_PRICE NUMBER(10,3)
);

-- *******************************************
--   Setup a publication
-- *******************************************


CREATE TABLE PUB_ORDER (
   ORDER_ID NUMBER,
   ORDER_DESCRIPTION VARCHAR2(128),
   ORDER_PRICE NUMBER(10,3),
   ADB_SUBJECT VARCHAR2(255) NULL,
   ADB_SEQUENCE INTEGER NOT NULL,
   ADB_SET_SEQUENCE INTEGER NULL,
   ADB_TIMESTAMP DATE NULL,
   ADB_OPCODE INTEGER NOT NULL,
   ADB_UPDATE_ALL INTEGER NULL,
   ADB_REF_OBJECT VARCHAR2(64) NULL,
   ADB_L_DELIVERY_STATUS CHAR NULL,
   ADB_L_CMSEQUENCE NUMBER(38, 0) NULL,
   ADB_TRACKINGID VARCHAR2(40) NULL
)
/
```

```
CREATE UNIQUE INDEX IDX1_PUB_ORDER ON PUB_ORDER (ADB_SEQUENCE)
/

CREATE UNIQUE INDEX IDX2_PUB_ORDER ON PUB_ORDER
(ADB_L_DELIVERY_STATUS, ADB_SEQUENCE)
/

CREATE INDEX IDX3_PUB_ORDER ON PUB_ORDER (ADB_L_CMSEQUENCE)
/

CREATE SEQUENCE PUB_ORDER_SEQ
   START WITH 1
   INCREMENT BY 1
   NOMAXVALUE
   NOCYCLE
   CACHE 10
/

CREATE OR REPLACE TRIGGER TRI_PUB_ORDER AFTER INSERT OR DELETE OR
UPDATE ON ORDER_TABLE
FOR EACH ROW
DECLARE
   updating_key_fields EXCEPTION;
BEGIN
   IF INSERTING THEN
   INSERT INTO PUB_ORDER VALUES (
      :NEW.ORDER_ID,
      :NEW.ORDER_DESCRIPTION,
      :NEW.ORDER_PRICE,
      NULL,
      PUB_ORDER_SEQ.NEXTVAL,
      0,
      SYSTIMESTAMP,
      1,
      NULL,
      NULL,
      'N',
      -1,
      NULL);
   END IF;
   IF UPDATING THEN
      IF UPDATING('ORDER_ID') THEN
         RAISE updating_key_fields;
      END IF;
   INSERT INTO PUB_ORDER VALUES (
         :OLD.ORDER_ID,
         :NEW.ORDER_DESCRIPTION,
         :NEW.ORDER_PRICE,
      NULL,
      PUB_ORDER_SEQ.NEXTVAL,
         0,
      SYSTIMESTAMP,
         2,
      NULL,
      NULL,
      'N',
      -1,
      NULL);
   END IF;
   IF DELETING THEN
   INSERT INTO PUB_ORDER VALUES (
      :OLD.ORDER_ID,
      :OLD.ORDER_DESCRIPTION,
      :OLD.ORDER_PRICE,
      NULL,
      PUB_ORDER_SEQ.NEXTVAL,
      0,
      SYSTIMESTAMP,
      3,
      NULL,
      NULL,
      'N',
      -1,
      NULL);
   END IF;
   EXCEPTION
      WHEN updating_key_fields THEN
         raise_application_error(-20300, 'ActiveDB Error: cannot
         update key fields of source table.');
END TRI_PUB_ORDER;
/

-- *******************************************
--   Setup a subscription
-- *******************************************


CREATE TABLE SUB_ORDER_EXCEP (
   ORDER_ID NUMBER,
   ORDER_DESCRIPTION VARCHAR2(128),
   ORDER_PRICE NUMBER(10,3),
   ADB_OPCODE INTEGER NULL,
   ADB_UPDATE_ALL INTEGER NULL,
   ADB_TRACKINGID VARCHAR2(40) NULL,
   ADB_JOIN_ID VARCHAR2(46) NULL,
   ADB_ERROR_TEXT VARCHAR(4000) NULL,
   ADB_ERROR_TIME DATE DEFAULT SYSDATE
)
/

COMMIT;
```

Incremental Parent-child Operation

To support incremental parent-child operations, each child row has an opcode, that is, an extra ADB_OPCODE field that is added to the child schema. The opcode ADB_OPCODE_BYPASS is used to bypass the current table operation. The adapter determines if the operation is an incremental parent-child operation by checking the first level child opcode. If the first level child opcode is not set, the adapter treats it as a complete operation. For the subsequence child level, if the child opcode is not set, it will inherit the parent opcode.

Mixed parent-child operations are also supported. You can send a message to insert new child rows, update other child rows, and delete other child rows for an existing parent-child object.

When working with parent and child tables, the value of ADB_OPCODE in the parent table and the child table must be the same. Or only set the value of ADB_OPCODE in the parent table and leave the child table empty.

Following is an example of a mixed parent-child operation:

```
adb.key
{
  RVMSG_INT      2  ^type^        1
  RVMSG_INT      2  ^pfmt^        10
  RVMSG_INT      2  ^ver^         30
  RVMSG_INT      2  ^encoding^    1
  RVMSG_RVMSG  110  ^prefixList^
  {
    RVMSG_STRING  49  1
"/tibco/public/sequence/ae/class/ae/ADB/abc"
    RVMSG_STRING  37  default  "/tibco/public/class/ae/ADB/abc"
  }
  RVMSG_RVMSG   77  ^tracking^
  {
    RVMSG_STRING  30  ^id^  "Gi2--4--DGMSk--s-064zzw8L-zzw"
    RVMSG_STRING  22  ^1^   "adb.key"
  }
  RVMSG_RVMSG  1200  ^data^
  {
    RVMSG_STRING    8  ^class^                "S_KEYP1"
    RVMSG_INT       4  ADB_OPCODE             10
    RVMSG_RVMSG   480  ADB_SEQUENCE_S_KEYP2
    {
      RVMSG_STRING   18  ^class^  "sequence[S_KEYP2]"
      RVMSG_INT       4  ^idx^    1
      RVMSG_RVMSG   210  ^1^
      {
        RVMSG_STRING    8  ^class^            "S_KEYP2"
        RVMSG_INT       4  ADB_OPCODE         10

        RVMSG_RVMSG   119  ADB_SEQUENCE_S_KEYP3
        {
          RVMSG_STRING  18  ^class^  "sequence[S_KEYP3]"
          RVMSG_INT       4  ^idx^    1
          RVMSG_RVMSG    58  ^1^
          {
            RVMSG_STRING  8  ^class^             "S_KEYP3"
            RVMSG_INT     4  ADB_OPCODE          4
            RVMSG_REAL    8  A                   1
            RVMSG_STRING  2  B                   "a"
            RVMSG_REAL    8  C                   4
          }
          RVMSG_RVMSG    58  ^2^
          {
            RVMSG_STRING  8  ^class^             "S_KEYP3"
            RVMSG_INT       4  ADB_OPCODE        3
            RVMSG_REAL    8  A                   1
            RVMSG_STRING  2  B                   "a"
            RVMSG_REAL    8  C                   4
          }
        }
      }
      RVMSG_RVMSG   210  ^2^
      {
        RVMSG_STRING    8  ^class^               "S_KEYP2"
        RVMSG_INT       4  ADB_OPCODE            1
        RVMSG_REAL      8  A                     2
        RVMSG_STRING    2  B                     "a"
        RVMSG_REAL      8  C                     4
        RVMSG_RVMSG   119  ADB_SEQUENCE_S_KEYP3
        {
          RVMSG_STRING  18  ^class^  "sequence[S_KEYP3]"
          RVMSG_INT       4  ^idx^    1
          RVMSG_RVMSG    58  ^1^
          {
            RVMSG_STRING  8  ^class^             "S_KEYP3"
            RVMSG_REAL    8  A                   2
            RVMSG_STRING  2  B                   "a"
            RVMSG_REAL    8  C                   4
          }
        }
      }
    }
  }
}
```

When the adapter receives this message, it performs the following database operations in sequence:

1. Bypass the parent table operation.
2. Bypass first row operation on child table `S_KEYP2`.
3. `UPDATE S_KEYP3 set B = 'a', C = 4 where A = 1;`
4. `DELETE FROM S_KEYP3 where A = 1;`
5. `INSERT INTO S_KEYP2 (A, B, C) values (2, 'a', 4);`
6. `INSERT INTO S_KEYP3 (A, B, C) values (2, 'a', 4);`

Remove Records from the Publication Table

An adapter instance does not remove the records from the publication table automatically, because that information must be retained for auditing reasons. You can use the following command at the SQL prompt to manually remove records from the given publishing table. The command deletes all rows where `ADB_L_DELIVERY_STATUS` is `C` or `F`.

```
SQL> delete from publication_table_name where ADB_L_DELIVERY_STATUS = "C"
or ADB_L_DELIVERY_STATUS = "F";
```

To automate the process, you can include this statement in a trigger.

Change the Publication Trigger to Publish a Subset of Rows

You can set up a publication to publish only a subset of the rows within a table. That is, there may be scenarios where it is desirable to publish only if the inserted, deleted, or updated row satisfies certain conditions. It is possible to use the callout library to do this additional filtering. It is simpler and more efficient to use a trigger that directly tests whether publishing conditions are met.

For Oracle, add a `when` clause to a row level trigger to test for the desired conditions. For Sybase and Microsoft SQL Server, use the `if` statement within a trigger body to test for the desired conditions.

For example, you can change the `demo1_ora.sql` trigger described in the previous section to fire only if the `ORDER_PRICE` value is `$1.00` or greater:

```
CREATE OR REPLACE TRIGGER TRI_P_ORDER_TABLE AFTER INSERT OR UPDATE
OR DELETE ON ORDER_TABLE
FOR EACH ROW WHEN (new.ORDER_PRICE >= 1.00)
```

Optimize the Publication Sequence for Oracle RAC System

To optimize the performance and to avoid out of order generation of **`ADB_SEQUENCE`**, it is recommended that you add the ORDER option and also increase the cache size.

Consult your system administrator to confirm the impact of these changes.

```
CREATE SEQUENCE PUB_ORDER_SEQ
   START WITH 1
   INCREMENT BY 1
   NOMAXVALUE
   ORDER
   NOCYCLE
   CACHE 50
```

Source Table

If loop detection is enabled, the following column will be added to the source table.

If the source table contains a field ADB_TRACKINGID, it will be updated using the tracking id of the message. If this field is not present, the tracking id field will be ignored..

Table 37 Source Table: Additional Columns

Column Name	Type	Description
ADB_SOURCE	CHAR	Used for loop detection. Denotes whether the row is inserted or updated as the result of a TIBCO Rendezvous message, rather than user intervention. Valid values are T or NULL. T indicates the row is not to be published. NULL indicates the row can be published.
ADB_TRACKINGID	VARCHAR2 (40)	Tracking ID of the message.

The source table cannot contain any user-created columns where the column name starts with ADB_. These characters are reserved for adapter use.

Specifying the Polling Method

An adapter Publication Service uses periodic polling or an alerter process to monitor changes to a database table. You can choose the method with the **Polling Method** list in the Publication Options tab.

- **Timer** This is the default polling method. The adapter polls (checks) the publishing tables periodically for any new rows to be published. This method is the most efficient when the publishing tables change frequently and a limited number of database operations are preferred.

  See Publishing Table on page 113 for details.

- **Alerter** In the case where the publishing tables change infrequently, polling may result in many unnecessary database accesses. For this situation, the alerter can be used to asynchronously alert the adapter of changes in the database. This removes the need for the adapter to poll its publishing table for existence of new rows. When the adapter receives an alert, it publishes the new data.

  The alerter method is only available when using an Oracle or a Microsoft SQL Server database.

  See Using an Alerter on page 123 for details.

Using a Timer

To use a timer to poll the changes in the database:

1. Select **Timer** from the **Polling Method** list in the Publication Options tab.
2. Specify a polling interval in milliseconds with the **Polling Interval** field. This is how often the service checks the publishing table for new rows.

See Publication Options Tab on page 103 for details of each field.

Using an Alerter

To use an alerter, select **Alerter** from the **Polling Method** list in the Publication Options tab, and you need to set up the database as well.

Alerter Introduction

The alerter process is used to asynchronously notify running Publication Service instances of database changes. The adapter does not need to poll its publishing table for existence of new rows in every interval. Use the alerter only when database changes are infrequent.

The procedures can be executed on the SQL command line or through any supported Application Programming Interfaces (the procedures cannot be invoked successfully from within a trigger). The procedures commit the inserts into the database table and notify the adapter of the commit.

Because the alerter process runs with a database instance, not user schema, an adapter instance cannot poll the publishing table name for just one user schema. For example, if the same publishing table exists for two user schemas and two instances are monitoring the same publication table, each instance with a different user, both instances will select the same publication table whenever the alerter checks for changes in the publication table.

Currently the alerter is only available in the following databases:

- Oracle

  An alerter for Oracle is part of the adapter and available on all operating systems supported by the adapter.

- Microsoft SQL Server or above

The alerter is supported on both the Rendezvous and JMS transport types.

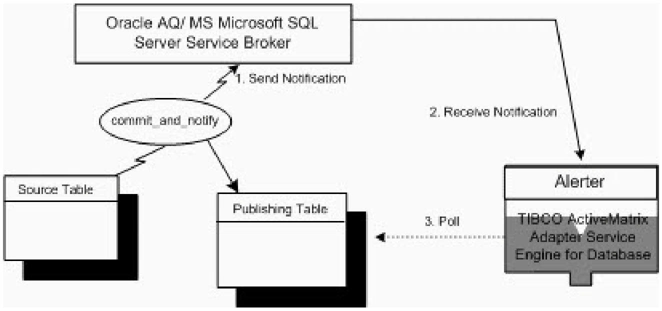
Set Up and Start the Alerter

The alerter on Oracle uses the Oracle AQ package and on MS SQLServer 2005 and above uses the Service Broker component to send and receive information between sessions, synchronously allowing two or more sessions in the same database instance to communicate.

The *Microsoft SQL Server* mentioned in this section refers to version 2005 and above.

As shown in Figure 3, when a source table is updated with data and the `commit_and_notify` procedure executed, a trigger copies the data to the publishing table and notifies the Oracle AQ or MS SQLServer, and that publishing table has changed. The alerter gets the notification from Oracle AQ or MS SQLServer and sends a notification message to the adapter that a publishing table has changed. The adapter then queries all its configured publishing tables for the new data and sends it on a subject to the TIBCO transport. One or more instances can be notified.

Figure 3 Alerter Operation (Oracle and MS SQLServer)


Oracle AQ/ MS Microsoft SQL Server Service Broker
1. Send Notification
commit_and_notify
2. Receive Notification
Source Table
Publishing Table
Alerter
3. Poll
TIBCO ActiveMatrix Adapter Service Engine for Database

If there are multiple publishing tables under the same database account, you can use the `commit_and_notify_table` procedure to specify that only a particular table be checked by the adapter. This prevents the adapter from needlessly checking all its publishing tables for updates when only one table has been updated with new data. This notification can be sent to one or more instances.

The following object types and stored procedures are available for committing messages when the alerter is used:

Table 38 Alerter Object Types and Stored Procedures

Name	Type	Description
adb_alerter_qtbl (Oracle only)	AQ queue table	Queue table for all alerters.
adb_alerter_q (Oracle only)	AQ queue	Queue for all alerters.
adb_alerter_typ	Service Broker Message Type	Object type for all messages.
adb_alerter_contract (Microsoft SQL Server 2005 only)	Service Broker Contract	The contract type used. The contract specifies the message types that can be used.
*instance_Id*_q (Microsoft SQL Server 2005 only)	Service Broker Queue	Created by the config_alerter stored procedure.
*instance_Id*_send_service (Microsoft SQL Server 2005 only)	Service Broker Service	Created by the config_alerter stored procedure.

Table 38 Alerter Object Types and Stored Procedures (Cont'd)

Name	Type	Description
*instance_Id*_rcv_service (Microsoft SQL Server 2005 only)	Service Broker Service	Created by the config_alerter stored procedure.
adb_alerter_typ	Object	Object type for the payload message.
commit_and_notify	Procedure	Sends a change notification to all adapters.
commit_and_notify_agent *alerter_name*	Procedure	Sends a change notification to the named alerter.
commit_and_notify_table *table_name*	Procedure	Sends a change notification for a named table.
stop_alerter	Procedure	Stops all running alerters.
stop_alerter_agent *alerter_name*	Procedure	Stops the alerter for the named alerter.
config_alerter	Procedure	Alerter call to register itself as an AQ subscriber.
cleanup_alerter	Procedure	Alerter call to unregister itself as an AQ subscriber.
listen_pipe	Procedure	Alerter call to block on alerts.

Before Starting

When using the alerter functionality with an Oracle database, make sure you have executed the `create_user.sql` and `alerter.sql` files with the database account used by the adapter, as described in Post-Installation Tasks in *TIBCO ActiveMatrix Adapter for Database Installation*.

When using the alerter functionality with a Microsoft SQL Server database, make sure you have enabled the service broker functionality in the database server. To check if the service broker is enabled, execute:

```
SELECT is_broker_enabled FROM sys.databases WHERE database_id =
DB_ID()
```

The returned value is `1`.

To enable service broker, execute

```
Alter database dbname set enable_broker with rollback immediate
```

Configure and Start the Alerter

To use the alerter, you must install the Oracle AQ or Microsoft SQL Server package on the database server side. See your Oracle documentation for information about installing the Oracle AQ or Microsoft SQLServer package. When configuring the Publication Service, you must enable the alerter by selecting the **Alerter** item from the **Polling Method** list in the Publication Options tab. When the adapter starts, the alerter also starts.

- When you select the alerter, the Polling Interval field will be disappeared.
- If an adapter that uses the alerter is not shutdown cleanly, you must call the `cleanup_alerter` stored procedure before restarting the adapter. The procedure is normally called by the adapter when it shuts down cleanly.

Oracle Alerter Example

This section describes how to execute the `commit_and_notify_table` stored procedure. It assumes the `rvpub.tra` configuration has its data source name, user account name (`demo`) and password defined in the repository. The `rvpub` adapter Publication Service has been configured with the Use Alerter check box selected.

1. Start the Publication Service instance by typing:

```
adbagent --propFile rvpub.tra
```

2. After inserting a message, execute the `commit_and_notify_table` stored procedure with the publishing table monitored by the adapter. For example, if the adapter is monitoring the `PUB_ORDER` table:

```
sqlplus demo/demo
insert into ORDER_TABLE values(111,'Oak Table',499.95);
SQL> call commit_and_notify_table ('PUB_ORDER');
```

The procedure puts a message on an Oracle AQ. The publisher adapter then reads its publishing table and sends a message containing the changed data on its configured subject.

This example shows how to tell the adapter instance to poll a single table, `PUB_ORDER`, for changes. Use the `commit_and_notify` stored procedure to poll all publishing tables for changes.

SQL Server Alerter Example

This section describes how to execute the `commit_and_notify_table` stored procedure. It assumes the `rvpub.tra` configuration has its data source name, user account name (`demo`) and password defined in the repository. The `rvpub` adapter Publication Service has been configured with the Use Alerter check box selected.

1. Start the adapter Publication Service instance by typing:

```
adbagent --propFile rvpub.tra
```

2. After inserting a message, execute the `commit_and_notify_table` stored procedure with the publishing table monitored by the adapter. For example, if the adapter is monitoring the `PUB_ORDER` table:

    ```
    isql -Udemo -Pdemo
    SQL> insert into ORDER_TABLE values(111,'Oak Table',499.95);
    SQL> execute commit_and_notify_table ’PUB_ORDER’;
    ```

    The procedure puts a message to the service broker queue, which is picked up by the adapter. The publisher adapter then reads its publishing table and sends a message containing the changed data on its configured subject.

    This example shows how to tell the adapter instance to poll a single table, `PUB_ORDER`, for changes. Use the `commit_and_notify` stored procedure to poll all publishing tables for changes.

Choosing Storage Mode

The adapter Publication Service provides two storage modes to copy the columns from the source table to the publishing table. The storage mode you choose depends on your specific requirements.

Publish by Value

With Publish by Value, all specified columns in the source table are copied to the publishing table. Publishing by Value is fast, but does not support some data types, for example, Oracle LONG and LONG RAW.

When you use Publish by Value, the following restrictions apply on publishing tables:

- Publishing tables cannot contain columns with LONG data types. If you have a source table that contains a column with a LONG data type, that column cannot be specified for inclusion in the publishing table. This is because the trigger generated by the palette cannot copy the LONG column value through the `:new` construct.

  This is an Oracle restriction documented in the *Oracle SQL Reference* manual. The problem is not detected by Oracle during trigger creation. However, when the trigger fires and it attempts to copy the LONG column value to the publishing table, the database connection will hang for some time and then eventually terminate.

- When you define parent-child relationships between tables, the publishing table that is created for a parent table cannot contain a column with a LONG data type. However, a child table can contain a column with a LONG data type. This is because data on child table rows is not copied using the `:new` construct.

- LONG RAW data is not allowed in the publishing table.

These restrictions do not apply to publishing tables when you publish by reference, and LONG or LONG RAW are non-key types.

In the following example, the publishing table `P_CUSTOMER` is created for the source table, `CUSTOMER`. When `CUSTOMER` is updated, the new data will be copied to `P_CUSTOMER`. The adapter will poll `P_CUSTOMER` and publish the new data.

In this example, loop detection is enabled. If a subscription exists that uses the same subject and `CUSTOMER` as the destination table, any changes to `CUSTOMER` will not be published repeatedly.

Publish by Reference

With Publish by Reference mode, only key column values are copied to the publishing table. The publishing by reference feature allows you to publish data directly from the source table without copying the data from the source table to a publishing table firstly. A trigger, stored procedure, and publishing table are created, but the publishing table contains the necessary adapter fields and only the key fields of the source table.

The advantage of this feature is that the data to be published is stored just once. Also, data types, such as Oracle LONG and LONG RAW are supported for publishing by reference.

A key column or substitute key column is required when publishing by reference, because the publishing table contains only key values. If no column is specified, the publication is not added.

To use a view or other database object as the source table, you can configure the adapter to publish by reference object, where key columns are stored in the publishing table and data to publish is selected from the reference object. For details, see Publish by Reference Object on page 130.

In the following example, the publisher adapter is configured to publish from the P_CUSTOMER table with a key field CUST_ID. The publishing table is created with the necessary fields and the CUST_ID field. When a row in the P_CUSTOMER table is modified, the trigger fires, populates fields and copies the CUST_ID value to the publishing table. When the adapter polls the publishing table, it detects the new row and selects from the P_CUSTOMER table by using the CUST_ID value found in the publishing table. Then the message is published.

Publish by Reference Object

When source data is stored in a view or other database object, you can publish this data using the publish by reference object feature. Publishing by reference object is an extension of the publish by reference feature.

In both cases, only key values from the source table are stored in the publishing table. The difference when publishing by reference object is when a row changes in the source table and the associated trigger fires, the adapter fetches data from the reference object, rather than the source table. The name of the reference object is stored in an ADB_REF_OBJECT column in the publishing table. Publishing by reference object is recommended when a view provides the most efficient access to source data, for example when many levels of nesting exist between a parent and child table.

To Configure a Publication Service that Publishes Data by Reference Object

Follow these steps to use the publishing by reference object feature for a Publication Service. The procedure uses the example tables introduced in the Getting Started Tutorial on page 15.

1. Add a source table for the Publication Service by following the procedures introduced in Specifying the Schema on page 109.
2. Click the Publication Options tab and specify the following options:
   a. Select **Publish by Reference** from the **Storage Mode** list.

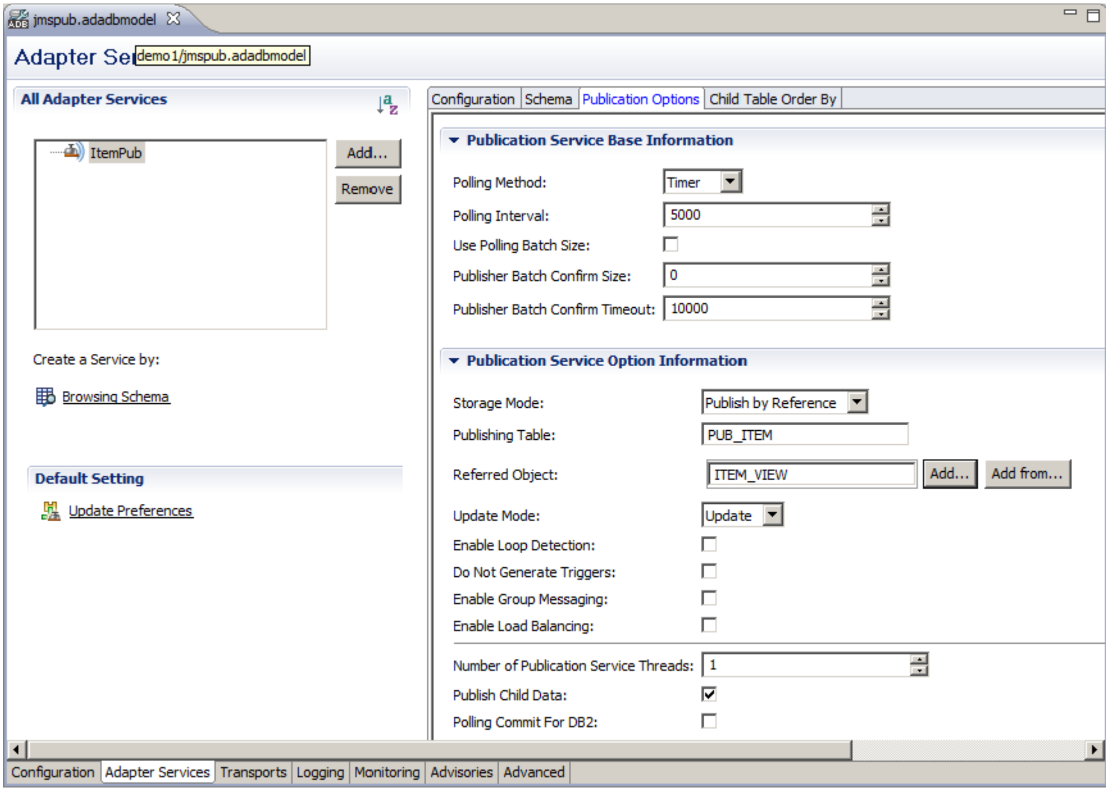
   b. Type the table name to use for storage in the Publishing Table field. A common practice is to use the source table name prefixed by **`P_`**.

   c. Type the name of the view or other database object to select source data from in the **Referred Object** field. To select from a list of tables in the current user schema, click **Add**. To select an object from a different schema, click **Add From**. For example, select **`ITEM_VIEW`**.

   d. Select the method for updating tables from the **Update Mode** list:

   - Select **Update** if you want to update a row in the destination table only if the row exists.

   - Select **Upsert** if you want to update a row in the destination table if the row exists and if no such row exists, perform an insert.



Publication Service Base Information

Polling Method: Timer

Polling Interval: 5000

Use Polling Batch Size: □

Publisher Batch Confirm Size: 0

Publisher Batch Confirm Timeout: 10000

Publication Service Option Information

Storage Mode: Publish by Reference

Publishing Table: PUB_ITEM

Referred Object: ITEM_VIEW

Update Mode: Update

Enable Loop Detection: □

Do Not Generate Triggers: □

Enable Group Messaging: □

Enable Load Balancing: □

Number of Publication Service Threads: 1

Publish Child Data: ☑

Polling Commit For DB2: □

3. Save the adapter configuration.

4. Designate a column of the reference object as the key.

   You must explicitly designate a key column or substitute key column for the reference object, because a reference object has no external designation of the key column:

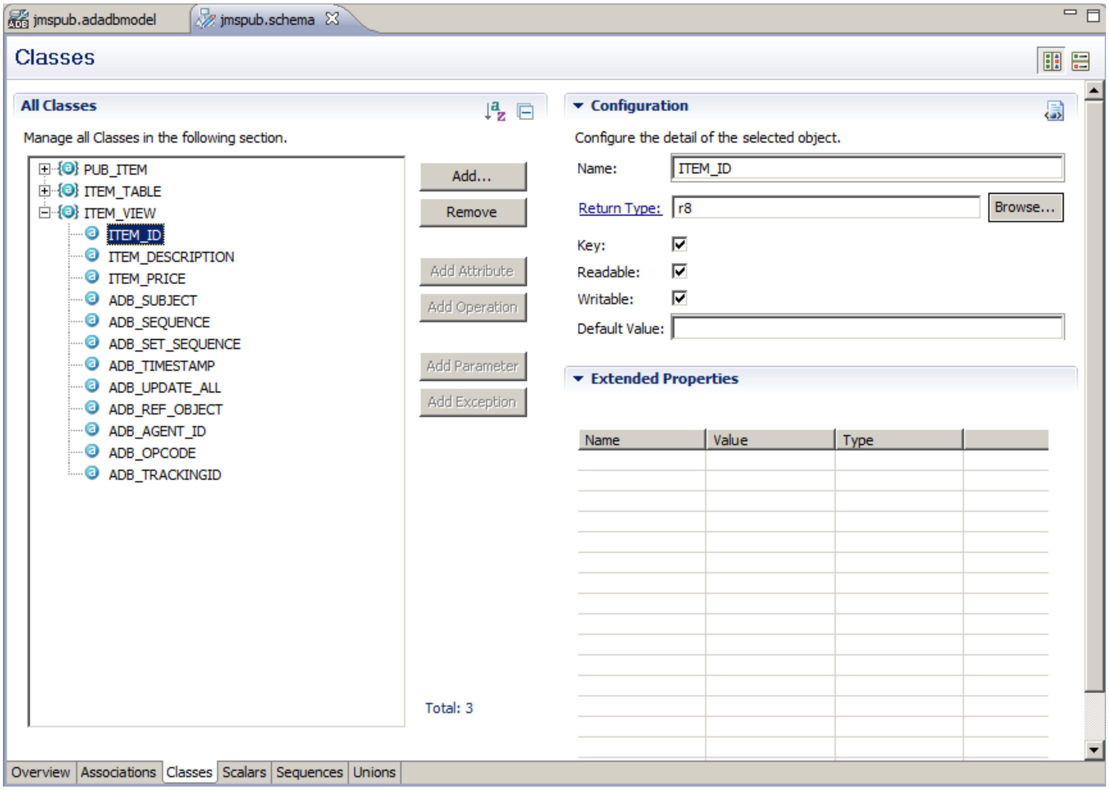
   a. Click the Configuration tab of the Publication Service.

   b. In the tab, click the **Class Reference** link in the Schema panel to open the AE Schema used by the adapter service.

   c. In the Classes tab, expand the reference object in the All Classes panel and select the column you want to designate as the key. For example, select **`ITEM_ID`**. Its configuration fields are displayed in the Configuration panel.

   d. From the Configuration panel, select the Key check box.

e. Save the AE Schema.



Classes

All Classes

Manage all Classes in the following section.

- PUB_ITEM
- ITEM_TABLE
- ITEM_VIEW
  - ITEM_ID
  - ITEM_DESCRIPTION
  - ITEM_PRICE
  - ADB_SUBJECT
  - ADB_SEQUENCE
  - ADB_SET_SEQUENCE
  - ADB_TIMESTAMP
  - ADB_UPDATE_ALL
  - ADB_REF_OBJECT
  - ADB_AGENT_ID
  - ADB_OPCODE
  - ADB_TRACKINGID

Total: 3

Configuration

Configure the detail of the selected object.

Name: ITEM_ID

Return Type: r8

Key: ☑

Readable: ☑

Writable: ☑

Default Value:

Extended Properties

Name	Value	Type	

To Configure Parent-Child Tables that Publish Data by Reference Object

If your Publication Service includes parent-child relationships, you need to also add a sequence and association to the metadata stored in the AE Schema. These tasks are described in the following sections. The procedures use the tables created in the parent-child example introduced in *TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) Examples*.

Task A Specify the Reference Object

You need to specify the reference object for the parent table and designate a column of the reference object as the key. Follow the procedure introduced in To Configure a Publication Service that Publishes Data by Reference Object on page 131. In this example, the parent table name is **`CUSTOMER`**. Its view, **`CUSTOMER_VIEW`**, is specified as the reference object.

Task B Add a Sequence

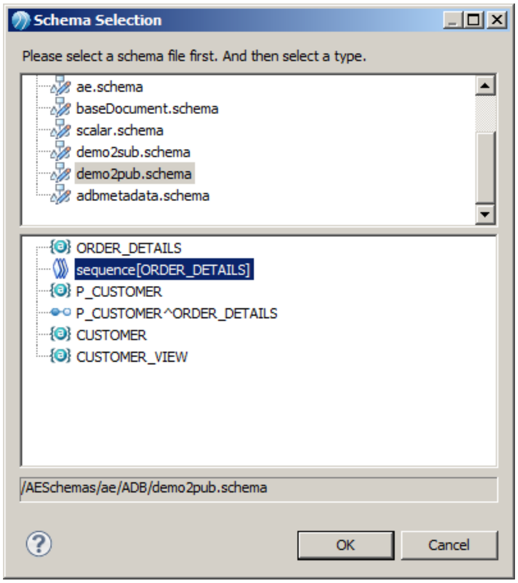
To add a sequence, follow these steps:

1. In the Configuration tab of the service, click the **Class Reference** link to open the AE Schema file in the editor.
2. Click the Classes tab, and select the reference object from the All Classes panel. For this example, select **`CUSTOMER_VIEW`**.

3. Click Add Attribute. In the Configuration panel, specify the following fields for the attribute:

   — **Name**: Type **ADB_SEQUENCE_*child_table_name*** in the Name field. For this example, the child table name is **ORDER_DETAILS**. So type **ADB_SEQUENCE_ORDER_DETAILS**.

   — **Return Type**: Click **Browse**. In the opened Schema Selection dialog, select the schema of the adapter configuration, for example, **demo2pub.schema**. In the displayed list, select the sequence for the child table, for example, **sequence[ORDER_DETAILS]**. Click **OK**.



Schema Selection

Please select a schema file first. And then select a type.

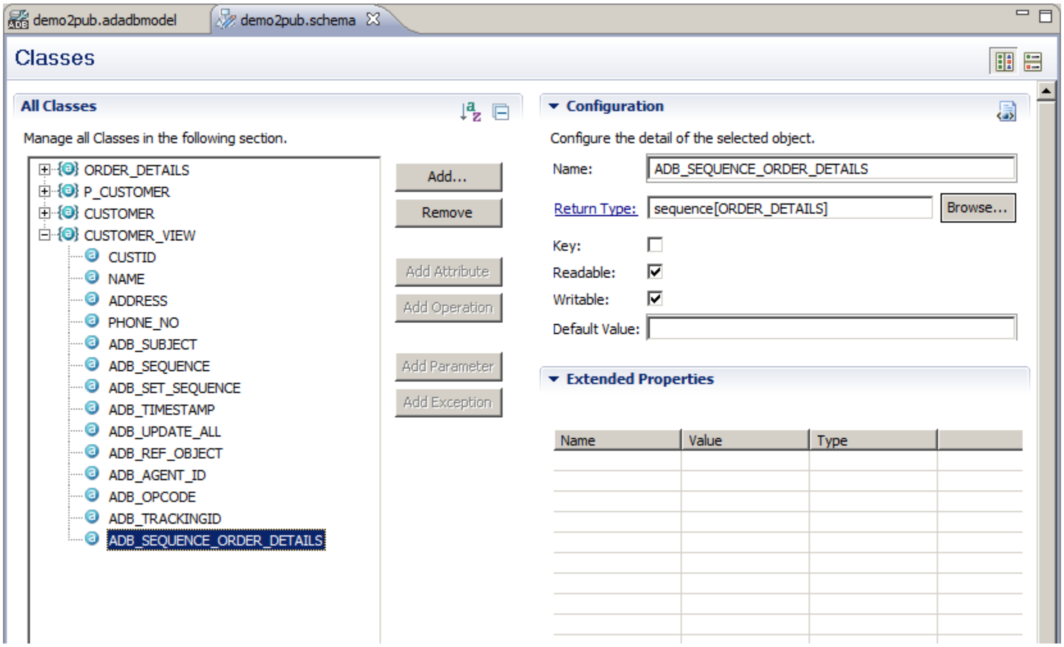
- ae.schema
- baseDocument.schema
- scalar.schema
- demo2sub.schema
- demo2pub.schema
- adbmetadata.schema

- ORDER_DETAILS
- sequence[ORDER_DETAILS]
- P_CUSTOMER
- P_CUSTOMER^ORDER_DETAILS
- CUSTOMER
- CUSTOMER_VIEW

/AESchemas/ae/ADB/demo2pub.schema

   — **Readable**: select the check box.

   — **Writable**: select the check box.



Classes

All Classes

Manage all Classes in the following section.

- ORDER_DETAILS
- P_CUSTOMER
- CUSTOMER
- CUSTOMER_VIEW
  - CUSTID
  - NAME
  - ADDRESS
  - PHONE_NO
  - ADB_SUBJECT
  - ADB_SEQUENCE
  - ADB_SET_SEQUENCE
  - ADB_TIMESTAMP
  - ADB_UPDATE_ALL
  - ADB_REF_OBJECT
  - ADB_AGENT_ID
  - ADB_OPCODE
  - ADB_TRACKINGID
  - ADB_SEQUENCE_ORDER_DETAILS

Configuration

Configure the detail of the selected object.

Name: ADB_SEQUENCE_ORDER_DETAILS

Return Type: sequence[ORDER_DETAILS] Browse...

Key: □

Readable: ☑

Writable: ☑

Default Value:

Extended Properties

Name	Value	Type	

4. Save the AE Schema. The sequence is added to the metadata for the reference object.

Task C Add an Association

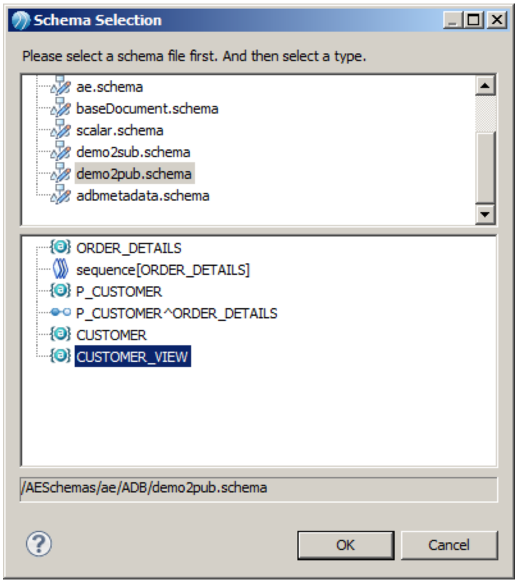
To add a sequence, follow these steps:

1. Still in the AE Schema editor, click the Associations tab.
2. Several associations are displayed in the All Associations panel, at least one for the parent table to the child table on the publisher side. Select this parent-child association, for example, **`P_CUSTOMER^ORDER_DETAILS`**.
3. In the Configuration panel, click in the **Name** field and replace the name of the publishing table with the name of the reference object. For example, **`P_CUSTOMER^ORDER_DETAILS`** becomes **`CUSTOMER_VIEW^ORDER_DETAILS`**.
4. Expand the parent-child association in the All Associations panel and select **Left Association**.

5. In the Configuration panel, specify the following fields and keep the rest fields unchanged:

   — **Name**: Replace the name of the publishing table with the name of the reference object. For example, `roleP_CUSTOMER` becomes `roleCUSTOMER_VIEW`.

   — **Endpoint Class**: Click **Browse**. In the opened Schema Selection dialog, select the schema of the adapter configuration, for example, `demo2pub.schema`. In the displayed list, select the reference object, for example, `CUSTOMER_VIEW`. Click **OK**.



Schema Selection

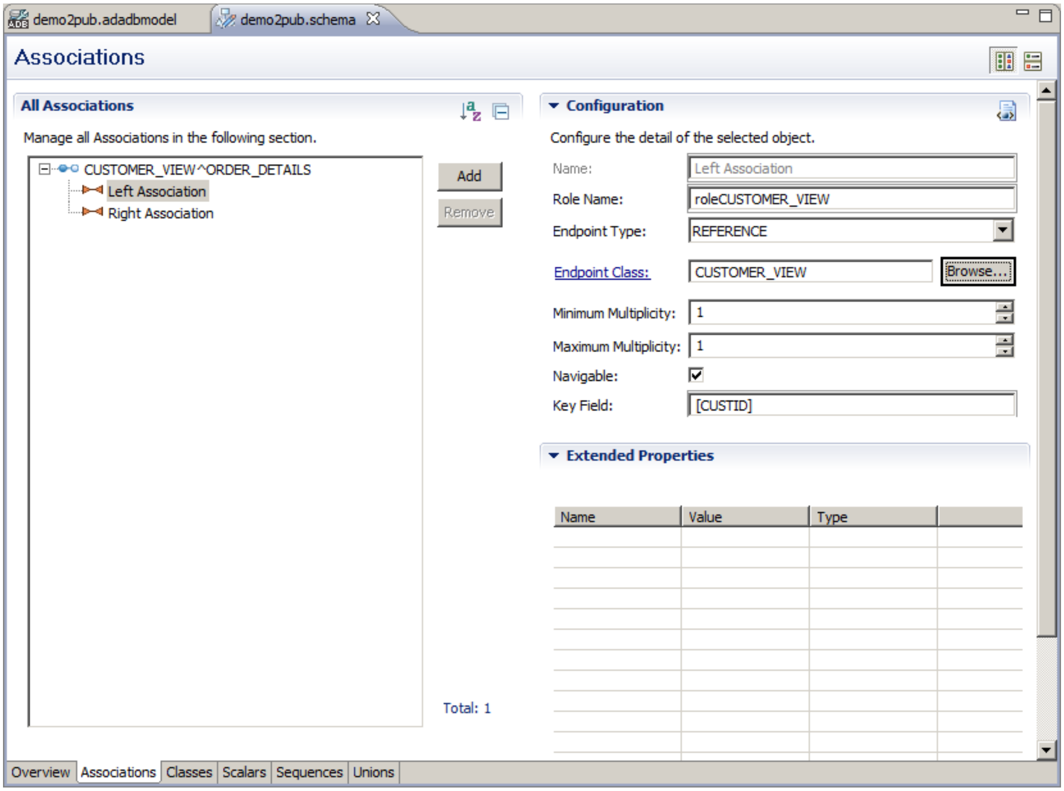
Please select a schema file first. And then select a type.

- ae.schema
- baseDocument.schema
- scalar.schema
- demo2sub.schema
- demo2pub.schema
- adbmetadata.schema

- ORDER_DETAILS
- sequence[ORDER_DETAILS]
- P_CUSTOMER
- P_CUSTOMER^ORDER_DETAILS
- CUSTOMER
- CUSTOMER_VIEW

/AESchemas/ae/ADB/demo2pub.schema

6. Save the AE Schema.



Name: Left Association
Role Name: roleCUSTOMER_VIEW
Endpoint Type: REFERENCE
Endpoint Class: CUSTOMER_VIEW Browse...
Minimum Multiplicity: 1
Maximum Multiplicity: 1
Navigable: ☑
Key Field: [CUSTID]

In this example, the Publication Service is configured to publish source data from the CUSTOMER table and its child table, ORDER_DETAILS. The relevant key columns are CUST_ID and ORDER_ID.

The publishing table is created with necessary adapter fields, as well as CUST_ID and ORDER_ID fields. When a row in CUSTOMER is modified, it fires populating fields and copying the CUST_ID value, as well as the name of the reference object, CUSTOMER_VIEW, to the publishing table. When the adapter polls the publishing table, it detects the new row. The adapter then selects the customer data from CUSTOMER_VIEW, using the CUST_ID value along with the view name found in the publishing table. Then the message is published.

Table to Record Sequence Numbers (DB2 on iSeries)

The palette will attempt to create the following table (if it does not exist) to record sequence numbers:

```
CREATE TABLE library.ADB_SEQTAB (
   PUB_TABLE VARCHAR(64) NOT NULL,
ADB_SEQ NUMERIC(20),
constraint ADB.ADB_SEQTAB_KEY primary key (PUB_TABLE))
```

The table must be journalled.

You can create this table manually. If journalling is automatically turned on, the palette will create the table automatically.

Enabling Loop Detection

If a source table is used both as the source and destination table for the same subject, you can enable the loop detection feature by select the check box in the Publication Options tab. The feature prevents the same changes from being published repeatedly.

If the loop detection feature is enabled, an additional column, `ADB_SOURCE`, is added to the source table. When an adapter receives a message, it inserts or updates the source table and enters a `T` in the `ADB_SOURCE` column to denote that this row was inserted or updated as a result of a message, rather than from user intervention.

Triggers created by the adapter are defined to not copy rows with `T` in the `ADB_SOURCE` column into the publishing table, which effectively means that the row will not be published. If you must update a row that was received (that is, has a `T` in the `ADB_SOURCE` column) and want the updated row propagated, you must change the `ADB_SOURCE` column to `NULL`, then the trigger will pick up the row and send it out.

Loop detection is disabled for DB2 on z/OS because DB2 on z/OS does not support drop column feature.

Master-Master Replication

You can use the loop detection feature to implement a simple master-master replication scheme. Master-master replication allows multiple sites, acting as peers, to copy and maintain groups of replicated objects.

When loop detection is enabled, an adapter can be configured as both a publisher and a subscriber to the same table on the same subject. When the subscriber receives a message, it compares the adapter ID of the message to its own adapter ID. If the values of the adapter ID match, and source and destination tables are the same table in the database, the subscriber discards the message. Other subscriber adapters listening on the subject receive the message only once.

The adapter does not resolve any replication conflicts, for example, two applications updating the same row of their corresponding source table will both publish the change. For advanced replication scenarios, use the replication tools provided by your database vendor.

Chapter 8 Configuring Subscription Services

A running Subscription Service automatically receives data from the messaging transport layer and update the database tables. In addition to its name, quality of service, and wire format, you can specify the reply options of the service. The adapter uses an exceptions table created in the database to store the error messages from database operation failures.

This chapter introduces the specific configuration for Subscription Services. For its general configuration, see Configuring the General Information of Adapter Services on page 41. For how to create an adapter service, see Creating Adapter Services on page 33.

Topics

Subscription Service Reference

Depending on the database you use, the Subscription Service options can be different.

Configuration Tab

Use this tab to configure the general information of the adapter service.

See Configuring the General Information of Adapter Services on page 41.

Schema Tab

Use this tab to specify the schema information of the adapter service. For how to specify the tables, see Specifying the Schema on page 148.

Tool Buttons

You can fetch and configure the tables by using the tool buttons in the Schema tab. The tables added are displayed in the Table panel.

- To add the root table: Click the **Add table** button. This table is the destination table.
- To add a secondary table to the source table: Click the **Add child table** button.
- To refresh stored table schema information by retrieving new information from the database, click the **Re-find tables from database** button.
- To delete a table, right-click the table and from the pop-up menu, click **Delete Table**. If the table has child tables, its child tables are also removed.

Schema Table

The following table lists and explains the table view of AE Schema.

Table 39 Subscription Service: Schema Tab—Schema Table View

Field	Description
Allow Key Columns Only	Use this check box to determines whether child columns can be joined to a column that is not key. • **Selected** Child columns are joined only to a key column. • **Cleared** Child columns can be joined to any column.
Tables and Columns	Loaded tables and columns.
Join To	Name of a parent table column to join to for parent-child relationships.
User Key	Click to define the column as a user key.
AE Type	Primitive type mapped to an TIBCO ActiveEnterprise type.

Subscription Options Tab

You can specify the following values in the Subscription Options tab:

Table 40 Subscription Service: Subscription Options Tab (Sheet 1 of 4)

Field	Description
Subscription Service Base Information	
Use Separate Session	This feature is used with Subscription Service multithreading. When selecting this check box, the Number of Subscription Service Threads field is displayed. After you set the value in this field, the multiple threads can share the same session and dispatcher. **Note:** When you need to use multiple threads by configuring a separate session, the transport session name must end with one or more digits from [0-9].

Table 40 Subscription Service: Subscription Options Tab (Sheet 2 of 4)

Field	Description
useSerial	Displays only when the Use Separate Session checkbox is selected. This option is designed for ensuring that all messages are executed in order. **Note:** • The primary key related in the destination table cannot be null. • If the destination table has no primary key, you need to select the User Key checkbox to instead of the primary key. When processing the application requests, the message is handled in order. • If the destination table has no primary key, and the User Key checkbox is cleared, then it is recommended to uncheck the useSerial checkbox.
Number of Subscription Service Threads	Displays only when the Use Separate Session checkbox is selected. Threads are allocated on demand. Valid values are from 1 through n. The number of thread you set in a Subscription Service indicates the number of the subscription thread that connects to the database. See Multithreading in Subscription Service on page 187 for more information.
Subscription Service Option Information	
Exceptions Table	Name of exceptions table where data is written if the adapter cannot write to the subscriber table. This table holds messages that caused an exception. If the table does not exist, the Subscription Service creates it. For details, see Exceptions Table on page 153. The exceptions table cannot contain any user-created columns where the column name starts with `ADB_`. These characters are reserved for use by the adapter. **Note:** For the parent-child relationship, this is the parent exceptions table. If you does not set the parent exceptions table, then the child exceptions table is invalid.
Use Opaque Exceptions Table	Select the check box to use an opaque exceptions table. The table records each message (entirely) into a column, along with the error message. A message is logged in the exceptions table if the Subscription Service fails to generate records in the destination table or the adapter fails to insert a message into an exceptions table. For details, see Opaque Exceptions Table on page 155. For DB2/390 databases, you must create a LOB tablespace before using the opaque exceptions table, which will use the LOB tablespace.
Opaque Exceptions Table	Name for the opaque exceptions table. For details, see Opaque Exceptions Table on page 155.
Pre Commit Stored Procedure	The value entered here represents the name of a stored procedure the subscriber will call after the database insert, update, or delete and prior to commit.

Table 40 Subscription Service: Subscription Options Tab (Sheet 3 of 4)

Field	Description
Reply Sender Quality of Service	If the subscriber must send a reply to the sender, this value identifies the quality of service or delivery mode to use when sending the reply. See Supported Sessions on page 46 and Guideline for Configuring the Delivery Mode (JMS Only) for a description of these fields. • Reliable (TIBCO Rendezvous transport type only) • Certified (TIBCO Rendezvous transport type only) • Persistent (JMS transport type only) • Non-persistent (JMS transport type only)
Bulk Insert Size	All incoming messages to insert are stored until this size is reached. Then, a bulk insert operation is performed on the destination table. This number must be less than or equal to the value in Batch Commit Size. The default value is `1`. **Note:** In the parent-child relationship situation, you need to pay more attention to the following: • Bulk Insert is valid if you set the group message and only send the data to the parent table. • Bulk Insert is invalid if you send the data to the parent table and child table. • Increasing Bulk Insert Size cannot significantly improve the adapter Subscription Service performance. • If you want to use JDBC driver options to improve the adapter Subscription Service Performance, TIBCO recommends that you can set `batchPerformanceWorkaround=true` to instead of `EnableBulkLoad and BulkLoadBatchSize` options.
Batch Commit Size	Number of messages to batch before invoking a commit operation. The default value is `0`. **Note:** • If you set the default value to `0` in the Batch Commit Size field, the adapter will take the value `1` as the batch commit size when running the Subscription Service. • Batch Commit option is not supported with RVCMQ Quality of Service. The RVCMQ scheduler requires a message to be confirmed before dispatching the next message, which prevents the adapter from operating in batch mode.
Batch Commit Timeout (milliseconds)	Specify an interval (in milliseconds) that can expire before confirmation messages regarding successful insertion into the exceptions table are sent back to the publisher. The default value is 10000 milliseconds. The batch commit feature does not commit all received messages if the adapter instance terminates before the batch commit value or time out value is met. When using RVCMQ, batch commit will timeout after each operation (insert/update/delete) when batch commit size>`1`.

Table 40 Subscription Service: Subscription Options Tab (Sheet 4 of 4)

Field	Description
Rendezvous Maximum Queue Size	Maximum number of messages to allow in the TIBCO Rendezvous event queue. The default value is 0, which means no limit is placed on event queue size. Use this option to prevent a subscriber's memory from overflowing if the publisher is too fast.

At startup, the Subscription Service dispatchers wait for all components to start before dispatching subscriber messages.

Child Table Mappings Tab

Child tables between a publisher and subscriber must be mapped unless the tables have the same name. Use this tab to complete mapping.

These options are only active when child tables have been specified on the Schema Tab.

You can specify values for the following fields in the Child Table Mappings tab. For how to map the child tables, see Set the Mapping between Child Tables on page 150.

Table 41 Subscription Service: Child Table Mappings Tab

Field	Description
Subscriber Child Table Name	Displays the child tables that can be mapped. Entries are automatically created when you add child tables for a Subscription Service.
Publisher Child Table Name	Click in this column and type the name of a publisher child table that corresponds to the subscriber child table name. Prefix the table name with a database user if your database requires it.

Child Exception Table Mappings Tab

You can configure each child table to use an exception table.

To make Child Exception Table Mappings effect, you need to set both the fields in the Child Exception Table Mapping tab and the exception table field in the Subscription Options tab.

You can specify values for the following fields in the Child Exception Table Mappings tab. For how to map exception tables with the child tables of the Subscription Service, see Specify the Child Exceptions Table on page 151.

Table 42 Subscription Service: Child Exception Table Mappings Tab

Field	Description
Subscriber Child Table Name	The child tables that can be mapped. Entries are automatically created when you add child tables for a Subscription Service.
Child Exception Table Name	Click in this column and type the name of an exception child table. **Note:** The child exception table is invalid if you does not set the parent exception table.

DB2/390 Options Tab

The DB2/390 Options tab is displayed, when you select **DB2 OS390** from the **Vendor** list in the Configuration tab of the adapter configuration.

Table 43 Subscription Service: DB2 OS390 Options Tab

Field	Description
Database Name	Name of the database that you want to put your subscriber table in.
Table Space Name	Name of the table space where the subscriber table is located.
LOB Table Space Name	Name of the LOB (Large Objects) table space name where auxiliary table of the opaque exceptions table is located.

Specifying the Schema

Each Subscription Service must be associated with the schema that describes the data the service sends to the database. You must specify the schema information before configuring other options for the service. The Schema tab of the service configuration is for specifying the schema. See Adding Child Tables on page 148 for the explanation of the schema table displayed in the tab.

Schema information includes tables and their columns. Use one of the following ways to specify the tables and columns:

- Use the tool buttons in the Table panel. See Tool Buttons on page 142 for details.
- Drag the fetched schema objects from the Schema Browser view to this tab. See Fetching Tables or Stored Procedures on page 80 for details.

You can configure an adapter to expect only a subset of the columns. Only subscribe columns that can be updated. See Restrictions of Subscribed Columns on page 152.

A Subscription Service can insert received data into parent and child tables. See Subscription Options Tab on page 143 for how to add related child tables in the service schema.

Destination table names can be qualified with a database schema name. The same as Publication Services, to access tables in other schemas, the database user specified in the adapter configuration must have the proper set of permissions granted. See Referencing an External Schema on page 83 for details.

If a primary key is not defined for a table, the update and delete triggers will not be generated for the table. To define a primary key for the table, select the User Key column in a table row. The update and delete triggers will then be defined.

Adding Child Tables

Data models typically contain tables that share column data through a relationship. When rows are inserted into a publishing table, a message that includes data from the source table and related (child) table is published. On the subscriber side, a corresponding table with the same columns as the child table associated with the publishing table must be specified.

Adding child tables requires two separate procedures, one for the publisher adapter and another for the subscriber. First, you add child tables for the source table, then you add child tables for the destination table. When you add a child table, TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) creates a class object in the repository for the child table, and an association object for the relationship.

This section explains the procedure of specifying child tables in Subscription Services. For the procedure of specifying child tables in Publication Services, see Adding Child Tables on page 109.

Restrictions of Child Tables

The following restrictions apply to parent and child tables for a Subscription Service configured to receive messages from a Publication Service:

- The child table in the source database and child table in the destination database must have the same columns, but the table names can be different. If the child table associated with the publishing table and the child table associated with the destination table have different names, you must set a mapping between the child tables when configuring the Subscription Service.
- When parent-child relationships are defined, a subscriber adapter must use the same repository as the publisher adapter.
- When working with parent-child table, it is recommended to set ADB_OPCODE in the parent table and child table to the same value. Or only set the value of ADB_OPCODE in the parent table and leave the child table empty.

When you add a child table, TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) creates a class object in the repository for the child table, and an association object for the relationship.

Procedure of Specifying Child Tables

To specify the child tables for the Subscription Service, complete the following tasks:

- Task A, Add Child Tables, page 150
- Task B, Specify the Relationship between the Parent and Child Tables, page 150
- Task C, Set the Mapping between Child Tables, page 150

See *TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) Examples* for how to configure an adapter configuration to publish and subscribe to data that is stored in related parent and child tables.

Task A Add Child Tables

To add child tables in the Schema tab, you can either use the tool buttons in the Schema tab, as explained in the following procedure, or drag the schema objects with child tables already added from the Schema Browser.

1. Click the name of the parent table and select it, then click the **Add child table** button.
2. In the Table name pattern dialog, specify the search criterion.
3. In the Select table dialog, select the related child table from the list.
4. (Optional) if you want to add tables from other schema, specify the schema name in the **Other Schema** field, and click **Add From Other Schema**, then select the related child table from the list.

   To reference an external schema, the user must have the proper access privileges. For instructions on granting access privileges to an external schema, see Referencing an External Schema on page 83
5. If you want to select the columns of the table to publish, click **Next** and select the check boxes next to the columns.
6. Click **Finish**.
7. Repeat step 1 to step 6 to add more child tables.

Task B Specify the Relationship between the Parent and Child Tables

You need to designate a foreign key column as a key in each child table so that a relationship to the parent table can be defined. You must also specify the relationship between the primary column in the parent table and the foreign key column in the child table.

Follow these steps:

1. Select the **User Key** check box for the foreign key column in the child table.
2. Click the **Join To** field of the child table column, and select the primary key column of the parent table from the list.

   Select the **Allow Key Columns Only** check box, child columns are joined only to a key column. Clear this check box, child columns can be joined to any column.

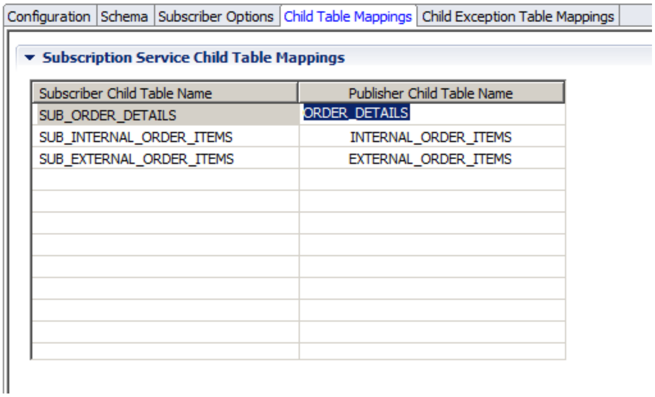
Task C Set the Mapping between Child Tables

Use the **Child Table Mappings** tab to set the mapping between the child tables in the Publication Service and the child tables in the Subscription Service.

The **Subscriber Child Table Name** column displays the child tables that can be mapped. Entries are automatically created when you add child tables for a Subscription Service. Click in the **Publisher Child Table Name** column and type the name of a publisher child table that corresponds to the subscriber child table name. Prefix the table name with a database user if your database requires it.

The following figure shows an example child table mapping.

Figure 4 Child Table Mappings


Configuration
Schema
Subscriber Options
Child Table Mappings
Child Exception Table Mappings
Subscription Service Child Table Mappings

Subscriber Child Table Name	Publisher Child Table Name
SUB_ORDER_DETAILS	ORDER_DETAILS
SUB_INTERNAL_ORDER_ITEMS	INTERNAL_ORDER_ITEMS
SUB_EXTERNAL_ORDER_ITEMS	EXTERNAL_ORDER_ITEMS

Task D Specify the Child Exceptions Table

You can configure each child table to use an exceptions table by using the **Child Exception Table Mappings** tab.

The **Subscriber Child Table Name** column displays The child tables that can be mapped. Entries are automatically created when you add child tables for a Subscription Service. Click in the **Child Exception Table Name** column and type the name of an exception child table.

The child exceptions table is invalid if you does not set the parent exceptions table.

To make the mapping of child exceptions tables effective, you need to set the mapping using the **Child Exception Table Mapping** tab as well as specify the **Exceptions Table** field in the Subscription Options tab.

Restrictions of Subscribed Columns

An incoming message need not contain data for all the columns defined in the subscriber table. You can configure a Subscription Service to expect only a subset of the columns. The service checks the repository for attributes defined in the class object definition of the subscriber table. When a message arrives, the service iterates through the attributes in the class object definition of the subscriber table and looks for those same attributes in the incoming message.

When configuring the destination table, only subscribe columns that can be updated. If you subscribe a column that cannot be updated and a message arrives with no data for that column, a `NULL` will be written to that column. For example:

- If a source table is configured to send data for columns, `c1`, `c2` and `c3`, and the destination table is configured to receive data for columns `c1`, `c2`, `c3`, `c4`, and `c5`:
    - — For the TIBCO Rendezvous message wire format, columns `c1`, `c2`, and `c3` will get the data and columns `c4` and `c5` will get a `NULL` value.
    - — For the TIBCO ActiveEnterprise message and XML message wire format, columns `c4` and `c5` will be ignored and take on whatever default values they are supposed to have.
- If a source table is configured to send data for columns, `c1`, `c2`, and `c3`, and the destination table is configured to receive data for columns `c1`, `c2`, and `c3` but not configured to receive data for columns `c4` and `c5`, columns `c4` and `c5` will retain the defaults applicable to both tables.

When the publisher table is configured to use parent-child relationships, the subscriber adapter must use the same repository as the publisher adapter.

Handling Exceptions

TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) uses exceptions tables and opaque exceptions tables for recording subscription failures. You can designate an exceptions table for a Subscription Service. If a subscriber adapter cannot write data to its destination table, it will write the data to the exceptions table.

Exceptions Table

If a database restriction or failure occurs, you can configure an exceptions table to receive the message. If insertion into an exceptions table also fails, an error message will display and the adapter instance will terminate.

You can build a TIBCO Hawk rulebase that detects when the configuration is down and automatically restarts it when the database is up. See *TIBCO Hawk Administrator's Guide* for details about creating a rulebase.

Specify the Exceptions Table

You can specify the exceptions table in the **Exceptions Table** field located in the Subscription Options tab.

When you specify an exceptions table, the adapter transform the table structure into an AE Schema. Click the **Class Reference** link in the Schema panel located in the Configuration tab to open the AE Schema. For more details about AE Schema, see Difference between Database Schema and TIBCO ActiveEnterprise Schema on page 76.

When you save the adapter configuration, the adapter creates the exceptions table in the database.

Exceptions Table Reference

In addition to destination table columns, the following columns are added to the exceptions table.

Table 44 Exceptions Table: Additional Columns

Column Name	Type	Description
ADB_OPCODE	NUMBER(38)	Operation code used by the adapter: 1 indicates INSERT 2 indicates UPDATE 3 indicates DELETE 4 indicates UPDATE if row exists, otherwise INSERT. If an incoming TIBCO Rendezvous message does not have an operation code, an INSERT occurs. **Note:** When working with parent-child table, the value of ADB_OPCODE in the parent table and the child table must be the same. Or only set the value of ADB_OPCODE in the parent table and leave the child table empty.
ADB_UPDATE_ALL	NUMBER(38)	Currently not used.
ADB_TRACKINGID	VARCHAR2(40)	Tracking ID of the message. This column is the primary key. Each exceptions table that is mapped to a child table is connected to the parent's exceptions table by this column
ADB_ERROR_TEXT	VARCHAR2(4000)	Text of the error from the database server or other source that caused the exception.
ADB_ERROR_TIME	DATE	Timestamp of the inserted record. For Oracle databases, the timestamp includes the `time zone` information.
ADB_JOIN_ID	VARCHAR(46)	Joined column used to link a parent record with its child record. ADB_JOIN_ID is generated from ADB_TRACKINGID and concatenated with the record number in the group. A child table's exceptions table is connected to its parent's table exceptions table by ADB_JOIN_ID.

The exceptions table cannot contain any user-created columns where the column name starts with `ADB_`. These characters are reserved for adapter use.

When using Teradata and PostgreSQL databases to work with the exceptions table, if the operation fails, all uncommitted data will be rolled back.

Child Exceptions Table Reference

In addition to child table columns, the following columns are added to a child exceptions table.

Table 45 Child Exceptions Table: Additional Columns

Column Name	Type	Description
ADB_ERROR_TEXT	VARCHAR	Text of the error from the database server or other source that caused the exception.
ADB_TRACKINGID	VARCHAR	TrackingID of the message.
ADB_JOIN_ID	VARCHAR	Tracking ID of the Message.

Use an Exceptions Table as a Source Table

If you want to publish from an exceptions table and also want to use that exceptions table as the source table, do not use the ADB_ERROR_TEXT or ADB_OPCODE column names. Instead, follow these guidelines:

- Create a database view that mirrors the exceptions table, but rename the ADB_ERROR_TEXT and ADB_OPCODE columns so that they do not begin with "ADB_."
- After renaming the columns, use the publishing by reference object feature (see Publish by Reference Object on page 130) and choose your view as the reference object.

Opaque Exceptions Table

The Subscription Service uses two logical layers when processing a message. The first layer decodes data from the message and the second layer provides the database transaction. If an exception occurs in the first layer, the adapter logs the message to the opaque exceptions table. In the second layer, if any DML command fails at any level, the adapter rolls back this transaction and starts another transaction, inserting into exceptions tables. If the insert into exceptions table transaction fails, the adapter then logs the message to the opaque exceptions table.

Specify the Opaque Exceptions Table

To specify the opaque exceptions table, follow these steps:

1. Click the Subscription Options tab of the adapter service.
2. Select the **Use Opaque Exceptions Table** check box.

3. Specify the table in the **Opaque Exceptions Table** field located in.

When you specify an opaque exceptions table, the adapter transform the table structure into an AE Schema. Click the **Class Reference** link in the Schema panel located in the Configuration tab to open the AE Schema. For more details about AE Schema, see Difference between Database Schema and TIBCO ActiveEnterprise Schema on page 76.

When you save the adapter configuration, the adapter creates the opaque exceptions table in the database.

Opaque Exceptions Table Reference

The opaque exceptions table records the entire message into a column along with the error message. The opaque exceptions table has the following columns:

Table 46 Opaque Exceptions Table Columns

Column Name	Type	Description
ADB_TRACKINGID	VARCHAR	Tracking ID of the message.
ADB_ERROR_TEXT	VARCHAR	Text of the error from the database server, Adapter SDK or other source that caused the exception.
ADB_ERROR_TIME	DATE	Timestamp of the inserted record. For Oracle databases, the timestamp includes the time zone information.
ADB_MSG	BLOB	Raw bytes of the message. **Note:** The default column size is 1 M. The Subscription Service will stop when the inserted data size is larger than the objective table column size and the opaque exceptions table column size.
ADB_SUBTAB	VARCHAR	Destination table name.
ADB_SUBJECT	VARCHAR	Subscription Service destination or subject.
ADB_TRANSPORT	INT	Subscription Service transport type: 0 unknown 1 Rendezvous 2 JMS

You can configure several Subscription Services in the `ADB_SUBTAB` column using only one opaque exceptions table in the same database schema.

Specifying a Pre-commit Stored Procedure Call

You can configure Subscription Service to call a stored procedure after the database insert, update, or delete and prior to commit. You can use this stored procedure as a hook to accomplish further processing inside the database and return the results to the adapter.

Specify the pre-commit stored procedure name in the **Pre-Commit Stored Procedure** field of the Subscription Options tab.

The adapter will call the pre-commit stored procedure with the following syntax:

```
{call <pre-commit stored procedure name>(?, ?, ?)}.
```

This stored procedure need to be defined with a specific interface as specified in the following table:

Table 47 Description of the Pre-commit Stored Procedure Parameters

Name	Type	Description
RETURN_ CODE	integer	`RETURN_CODE = 0`, the adapter assumes the procedure was successful and writes a success message to the SDK INFO trace role when the verbose mode is used. `RETURN_CODE <> 0`, the adapter assumes the procedure was not successful and writes `SP_TEXT` to the SDK ERROR trace role (whether verbose mode is on or `off`). If the message has a reply subject, this output value will be returned to the message sender. See Subscriber Replying Sender on page 158 for more details.
SP_TEXT	varchar	If the message has a reply subject, this output string will be returned to the message sender. See Subscriber Replying Sender on page 158 for more details.
DO_ROLL BACK	integer	`DO_ROLLBACK = 0`, the adapter commits the transaction and confirms the original message. `DO_ROLLBACK <> 0`, the adapter rollbacks the transaction and does not confirm the message. Note, not confirming the message changes the RVCM behavior. All subsequent RVCM messages will not be confirmed. It can only be done when this RVCM behavior is really desired.

Subscriber Replying Sender

The subscriber can be configured to send the subscription status as a reply to the message sender if the message contains a reply subject. The ADB_SUBSCRIBER_STATUS adapter schema is used for this reply message:

Table 48 Description of ADB_SUBSCRIBER_STATUS Class Schema

Field Name	AE Type	Description
RETURN_CODE	i4	return_code=0, when the data is inserted into the destination table successfully. return_code=-1, when the data is inserted into the exceptions table. return_code=5, when the data is failed to insert in both destination and exceptions tables. If you specify subscriber pre-commit stored procedure, this field will contain the value of the RETURN_CODE output parameter.
ADB_TEXT	string	Contains the status or error message of the subscriber operation.
CALLOUT_TEXT	string	Contains the error message from the database driver in case of error operation. If you specify subscriber pre-commit stored procedure, this field will contain the value of the SP_TEXT output parameter.

Chapter 9 Configuring Request-Response Services

A running Request-Response Service automatically receives requests from other applications, parses them, and translates them to queries to the database to retrieve data. The output data are wrapped in a schema and sent back to the caller. In addition to its name, quality of service, and wire format, you can specify the service mode, the invocation approach, performance tuning, and other options for the service.

This chapter introduces the specific configuration for Request-Response Services. For its general configuration, see Configuring the General Information of Adapter Services on page 41. For how to create an adapter service, see Creating Adapter Services on page 33.

Topics

Introduction to Request-Response Service

Request-Response Service often acts as a Request Reply Server or RPC (Remote Procedural Call) Server. Using a Request-Response Service, client applications can send SQL statements, stored procedures, or stored functions on a specified subject to an adapter configuration. A request may be a query to the database or any DDL or DML command to be performed on the database. The adapter processes the requests and returns the results in a response to the client. The response consists of a result code or one or more result sets, based on the request. A response can also be an error code or error description, if the request was not successful.

Multiple adapter configurations can be configured so that any one of the configurations receives and processes the request. For multiple configurations, you can use TIBCO Rendezvous Distributed Queue or JMS Queue to balance the load across a number of configurations in a queue. The task will be assigned to the least loaded member of the queue. See Load Balancing in Subscription Services and Request-Response Services on page 194 for details.

Request-Response Service Reference

Use the Request-Response Options tab to configure specific settings for Request-Response Services.

Configuration Tab

Use this tab to configure the general information of the adapter service.

See Configuring the General Information of Adapter Services on page 41.

Schema Tab

This tab displays the stored procedure information of the adapter service.

See Fetching Stored Procedures on page 82.

Request-Response Options Tab

You can specify values for the following fields in the Request-Response Options tab:

Table 49 Request-Response Service: Request-Response Options tab

Field	Description
Request-Response Service Base Information	
Mode	Specify an endpoint that is a Request Reply Server or RPC (Remote Procedural Call) Server. • Request Reply: A subscriber endpoint is created. This subscriber listens to request and publishes the reply to the reply subject. • RPC: A server endpoint is created. This server receives a request from and sends back the reply to the client.

Table 49 Request-Response Service: Request-Response Options tab (Cont'd)

Field	Description
Maximum Rows	Specify the maximum number of rows to fetch in the service level. This can be used to limit the memory usage of the adapter. The rows not fetched will be ignored by the adapter. **Note:** You can also set this field in the operation level when using Request-Response Service of RPC Mode with **Use Custom operations** check box selected, and the Maximum Rows value set in the operation level takes higher precedence over the value set in service level. Follow these steps to complete the configuration: 1. In the Configuration tab of the adapter service, click the **Class Reference** link. 2. In the opened AE Schema, click the Classes tab. 3. In the All Classes panel, click ***procedure_aeclass_name* > user_specified_procedure_name > REQUEST**. Click the **Type** link in the Configuration panel. 4. In the opened REQUEST schema, click the Classes tab. 5. In the All Classes panel, click **INPUT_OPTIONS > MAXROWS**. Specify the **Default Value** field in the Configuration panel. In the log file, only the maximum number of rows in the service level is printed.
Use Custom Operations	This check box is displayed when the **Mode** field is set to **RPC**. Select this check box and fetch the stored procedure by following the steps in Fetching Stored Procedures on page 82.
Use Separate Session	Select this check box to create a new session, and move the service endpoint to this session. This feature is used with Request-Response Service multithreading. When selecting this check box, the Number of Request-Response Service Threads field is displayed. After you set the value in this field, the multiple threads can share the same session and dispatcher. See Multithreading in Request-Response Service on page 188 for details. **Note:** When you need to use multiple threads by configuring a separate session, the transport session name must end with one or more digits from [0-9].
Number of Request-Response Service Threads	Indicate the number of threads the adapter service uses to connect to the database. Valid values are from 1 through n. Each thread has a separate connection to the database. Specifying multiple threads allows you to balance the incoming RPC request load. See Multithreading in Request-Response Service on page 188 for details.
Reply Subject	(Request-Reply mode only) Type a subject name that the adapter uses to respond, if no response, subject is specified in the request message. The subject name in the **Reply Subject** field cannot be same as the name in the **Destination** field.

Choosing the Service Mode

The adapter Request-Response Service supports two service modes: Request-Reply mode and RPC mode. The service mode you choose depends on your specific requirements.

Each Request-Response Service must be associated with the schema that describes SQL statements, stored procedures, functions, or packages the service sends to the database. The schema format depends on the mode you choose. See the corresponding section for each mode for details.

Schema Reference

The following table lists and explains the tables and columns displayed in the AE Schema table.

Table 50 Request-Response Service: Schema Tab—Schema Table

Field	Description
Classes and Fields	Loaded classes and fields.
AE Type	Primitive type of the AE Schema class or field.

Using Request-Reply Mode

Request-Reply mode can be regarded as a Subscription Service with SQL statement based schema. In Request Reply, a subscriber endpoint is created. This subscriber listens to request and publish the response to the reply subject.

TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) supports the following message formats for Request-Reply mode:

- Rendezvous Message (TIBCO Rendezvous transport type)
- XML Message (TIBCO Rendezvous transport type or JMS transport type)
- ActiveEnterprise Message (TIBCO Rendezvous transport type)

Requests

A request can contain one or more SQL statements, stored procedures, functions, or packages to be executed as a transaction. The text of the SQL statement follows the conventions for JDBC SQL syntax. All SQL statements supported by the DBMS are allowed and placeholders (represented by a question mark, '?') are permitted in the SQL statement, conforming to the JDBC rules. For performance reasons, it is recommended to use a SQL statement. The '?'convention should only be used to bind binary data or call stored procedures.

The adapter listens on the subject on which the application sends requests. The application uses TIBCO Rendezvous or the TIBCO Adapter SDK to send a self-describing message containing a request on the agreed subject to the adapter configuration.

The following are supported:

- DDL and DML SQL statements.
- The column size of the return resultset cannot exceed 128 characters.
- Multiple statements or procedures in a single transaction that send a nested TIBCO Rendezvous message.
- `rv_Send()`, `rv_SendWithReply()`, and `rv_Rpc()` calls.
- TIBCO Rendezvous Java APIs.

If `rv_SendWithReply()` or `rv_Rpc()` calls are used, the adapter sends the response to the Inbox subject that was set by the application for the response. Otherwise, the adapter sends the response on the response subject that was set at configuration time.

TIBCO ActiveEnterprise or XML Message Request Format

In these formats, the input class is SQL_REQUEST. The SQL_REQUEST class is described below. Also see SQL_STATEMENT Class on page 178.

```
<object name="SQL_REQUEST" lastModified="1036435805361" id="503">
  <assoc name="attribute">
    <string name="name" value="STATEMENTS"/>
    <ref name="attributeType"
value="/tibco/public/sequence/ae/class/ae/ADB/adbmetadata/sequence
[SQL_STATEMENT]"/>
    <string name="isKey" value="false"/>
    <string name="isReadable" value="true"/>
    <string name="isWriteable" value="true"/>
  </assoc>
  <assoc name="attribute">
    <string name="name" value="CLOSURE"/>
    <ref name="attributeType" value="/tibco/public/scalar/ae/any"/>
    <string name="isKey" value="false"/>
    <string name="isReadable" value="true"/>
    <string name="isWriteable" value="true"/>
  </assoc>
  <string name="family" value="ae"/>
  <string name="objectType" value="class"/>
</object>
```

TIBCO Rendezvous Message Request Format

This is an example structure of the nested self-describing request message sent by an application to the adapter.

Request

`{`

`rv_Name = “closure”, rvmsg_Type = RVMSG_OPAQUE, rvmsg_Data =` *optional_closure_data*

`rv_Name = “stmt”, rvmsg_Type = RVMSG_RVMSG, rvmsg_Data =` *Statement*

`rv_Name = “stmt”, rvmsg_Type = RVMSG_RVMSG, rvmsg_Data =` *Statement*

`rv_Name = “stmt”, rvmsg_Type = RVMSG_RVMSG, rvmsg_Data = Statement`

`. . .`

`}`

The `closure` field is an optional field. If included in the request, the response returns the same closure argument untouched, so the client application can use this information as a means of matching requests with responses.

The value *Statement* is a TIBCO Rendezvous Message of the following structure:

Statement

`{`

`rv_Name = “sql”, rvmsg_Type = RVMSG_STRING, rvmsg_Data =` *SQL_statement_with_possible_bind_variables*

`rv_Name = “maxrows”, rvmsg_Type = RVMSG_INT, rvmsg_Data =` *optional_max_number_of_rows_to_fetch*

`rv_Name = “bind”, rvmsg_Type = RVMSG_RVMSG, rvmsg_Data =` *Bind_data*

`rv_Name = “bind”, rvmsg_Type = RVMSG_RVMSG, rvmsg_Data =` *Bind_data*

`rv_Name = “bind”, rvmsg_Type = RVMSG_RVMSG, rvmsg_Data =` *Bind_data*

`. . .`

`}`

The value *Bind_data* is a `TIBCO Rendezvous Message` with the following structure:

Bind_data

`{`

`rv_Name = "position", rvmsg_Type = RVMSG_INT, rvmsg_Data =` *position_of_placeholder_starting_with_1_from_left_to_right*

`rv_Name = "column", rvmsg_Type = RVMSG_STRING, rvmsg_Data =` *in_format_**table-name.column-name**_whose_column_type_matches_this_bound_variable*

`rv_Name = "data", rvmsg_Type =` *type_of_bound_data*, `rvmsg_Data =` *value_of_bound_data*

`}`

If the `position` field is not specified, the order received is used.

Example TIBCO Rendezvous Message Requests

The section contains example requests that are based on SQL statements made by a client application and sent to an adapter instance for processing.

- **Simple SQL Request Statement**

  Given that a client application submits the following SQL statement:

  ```
  INSERT INTO order_table (order_id, order_description,
  order_price) values (1, ‘Order 1’, 1.10)
  ```

  The corresponding request could be formatted as a TIBCO Rendezvous Message similar to the following:

  ```
  {
   name = “closure” value = optional_closure_argument
   name = “stmt” value =

    {
  ```

```
   name = “sql” value = “INSERT INTO order_table (order_id,
order_description, order_price) values (1, ‘Order 1’, 1.10)”
   }
}
```

- **Bind Request Statement**

  Assume that a client application submits the following SQL statement, where `?` is a placeholder for an `adbDateTime` value:

  ```
     INSERT INTO order_table (order_id, order_date) values (2, ?)
  ```

  The corresponding request can be formatted as a message in `TIBCO Rendezvous Message` format similar to the following:

  ```
  {
   name = “closure” value = optional_closure_argument
   name = “stmt” value =
     {
     name = “sql” value = “INSERT INTO order_table (order_id,
   order_date) values (2, ?)”
     name = “bind” value =
        {
        name = “order_table.order_date” type = RVMSG_STRING value
   = “1999-08-20 10:20:01”
        }
     }
  }
  ```

  To create a bind parameter for binary data, the actual data is sent as an `RVMSG_OPAQUE`.

Not every data value sent to an SQL statement must be bound. The client can embed all these values into the actual text of the statement instead. Embedding the values directly into the SQL text results in better performance. However, binary values must be bound. Types such as `adbDateTime` can either be bound or included in the SQL text by using the native DBMS date conversion function.

For example, in ORACLE, you can use the `TO-DATE` function to enter a date:

```
INSERT INTO order_table (order_id, order_date) values (2,
TO_DATE(’1999-08-20 10:20:01’, ’YYYY-MM-DD HH24:MI:SS’));
```

- **Using Bind Entries**

  Bind entries are specific only to the `stmt {}` that contains them. For example, if you want to insert three rows into `REPLYTEST`, you need to have three `stmt {}` blocks (and thus, three sets of `bind` statements):

```
request: {
   stmt={ sql="INSERT INTO REPLYTEST (id, timestamp, bincol)
VALUES (5, ?, ?)" bind={ column="REPLYTEST.TIMESTAMP"
data="1991-10-10 12:10:10"} bind={ column="REPLYTEST.BINCOL"
data=[3 opaque bytes]}}
   stmt={ sql="INSERT INTO REPLYTEST (id, timestamp, bincol)
VALUES (5, ?, ?)" bind={ column="REPLYTEST.TIMESTAMP"
data="1991-11-10 12:10:10"} bind={column="REPLYTEST.BINCOL"
data=[4 opaque bytes]}}
   stmt={ sql="INSERT INTO REPLYTEST (id, timestamp, bincol)
VALUES (5, ?, ?)" bind={ column="REPLYTEST.TIMESTAMP"
data="1991-12-10 12:10:10"} bind={ column="REPLYTEST.BINCOL"
data=[5 opaque bytes]}}}
```

Responses

A response from the adapter to a client application has a result code and one or more result sets. Each result set contains nested self-describing messages, each of which encodes a result row, such as that returned from a query. A response can also return an error code and error description if the request was not successful.

TIBCO ActiveEnterprise or XML Message Response Format

In the TIBCO ActiveEnterprise Message or XML format, the default output class is SQL_BATCHRETURN. This class is described in SQL_ BATCHRETURN Class on page 177.

TIBCO Rendezvous Message Response Format

If `rv_SendWithReply()` or `rv_Rpc()` is used to send the request, the response is sent back on the subject or Inbox that was associated with the request.

If `rv_Send()` is used to send the request, because no response subject was specified with the request, the response subject set when the adapter was configured is used. To receive a response from the adapter, the client application must listen on the response subject.

The structure of a response message in TIBCO Rendezvous Message format is shown below.

```
Reply
{
rv_Name = “status” rvmsg_Type = RVMSG_INT rvmsg_Data = 0
rv_Name = “results” rvmsg_Type = RVMSG_RVMSG rvmsg_Data = Result
rv_Name = “closure”, rvmsg_Type = RVMSG_OPAQUE, rvmsg_Data =
optional_closure_data
}
```

where *Result* is a message in TIBCO Rendezvous Message format of the following structure:

```
Result
{
name = “row” type = RVMSG_RVMSG value = List_of_columns
name = “row” type = RVMSG_RVMSG value = List_of_columns
name = “row” type = RVMSG_RVMSG value = List_of_columns
. . .
}
```

where *List_of_columns* is a message in TIBCO Rendezvous Message format of the following structure:

```
List_of_columns
{
rv_Name = column_name, rvmsg_Type = type_of_bound_data, rvmsg_Data =
value_of_bound_data
rv_Name = column_name, rvmsg_Type = type_of_bound_data, rvmsg_Data =
value_of_bound_data
rv_Name = column_name, rvmsg_Type = type_of_bound_data, rvmsg_Data =
value_of_bound_data
. . .
}
```

If the request processing was not successful, the response can also return an error code and error description as shown next:

```
Reply
{
rv_Name = “status” rvmsg_Type = RVMSG_INT rvmsg_Data = nonzero_number
rv_Name = “sql” rvmsg_Type = RVMSG_STRING rvmsg_Data =
SQL_statement_that_caused_error
rv_Name = “error” rvmsg_Type = RVMSG_STRING rvmsg_Data = error_text
rv_Name = “closure”, rvmsg_Type = RVMSG_OPAQUE, rvmsg_Data =
optional_closure_data
}
```

The `status` value is an integer specifying success or error. Possible values are:

```
0:  ok                 // No error
1:  noMem              // Out of Memory
2:  notInitialized     // Object never initialized
3:  typeConversion     // Type conversion error
```

```
4:  dbNotFound       // Database not registered
5:  serverError      // Error reported by server
6:  serverMessage    // Message from server
7:  vendorLib        // Error in vendor's library
8:  notConnected     // Lost connection
9:  endOfFetch       // End of fetch
10: invalidUsage     // invalid usage of object
11: columnNotFound   // Column does not exist
12: invalidPosition  // invalid positioning within
                        object,i.e.bounds err
13: notSupported     // Unsupported feature
14: nullReference    // Null reference parameter
15: notFound         // Database Object not found
16: missing          // Required piece of information is missing
17: noMultiReaders  // This object cannot support multiple readers
18: noDeleter        // This object cannot support deletions
19: noInserter       // This object cannot support insertions
20: noUpdater        // This object cannot support updates
21: noReader         // This object cannot support readers
22: noIndex          // This object cannot support indices
23: noDrop           // This object cannot be dropped
24: wrongConnection  // Incorrect connection was supplied
25: noPrivilege      // This object cannot support privileges
26: noCursor         // This object cannot support cursors
27: cantOpen         // Unable to open
28: applicationError // For errors produced at the application
                        level
29: notReady         // For future use
```

Example TIBCO Rendezvous Message Responses

This section has two examples. The first shows a successful query response, and the second shows an unsuccessful query response.

Query Example

As an example, assume the SQL statement, SELECT * FROM ORDER_TABLE produces the following result:

```
SQL> SELECT * FROM ORDER_TABLE;

ORDER_ID    ORDER_DESCRIPTION       ORDER_PRICE
   1          Order 1                 1.00
   2          Order 2                 2.00
```

In this case, the self-describing message in TIBCO Rendezvous Message format produced by the adapter has the following structure:

```
{
name = “closure” value = optional_closure_argument
name = “status” value = 0
name = “results” value =
      {
      name = “row” value =
         {
         name = “order_id” type = RVMSG_INT value = 1
         name = “order_description” type = RVMSG_STRING value =
         “Order 1”
         name = “order_price” type = RVMSG_REAL value = 1.00
         }
      name = “row” value =
         {
         name = “order_id” type = RVMSG_INT value = 2
         name = “order_description” type = RVMSG_STRING value =
         “Order 2”
         name = “order_price” type = RVMSG_REAL value = 2.00
         }
   }
}
```

Error Example

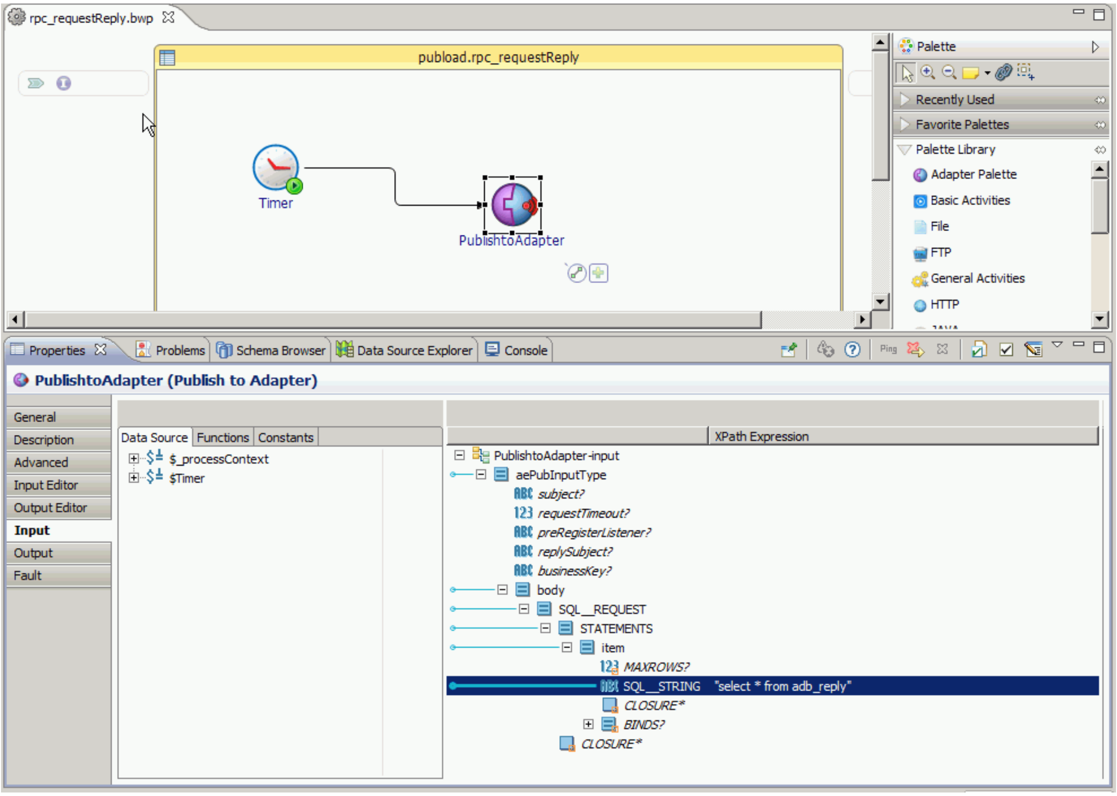
If the request that was processed resulted in an error, the adapter returns a message containing an error code and description as shown below. In this example, assume the SQL statement `SELECT * FROM ORDER_TABLE` produced an error because there was no table named `ORDER_TABLE` defined in the database:

```
SQL> SELECT * FROM ORDER_TABLE;
SELECT * FROM ORDER_TABLE
              *
ERROR at line 1:
CSHIONG2.ORDER_TABLE IS AN UNDEFINED NAME
```

Figure 5 indicates the input when using BusinessWorks to process the Request Reply operation.

When calling a Request-Response Service, the CLOSURE and DATA elements of anyType AE data type are used in the Adapter activity. You need to manually substitute anyType with a data type that matches the database metadata. See Handling the anyType AE Data Type on page 90 for the detailed information.

Figure 5 Input of the Request Reply Operation Using BusinessWorks



SQL__STRING "select * from adb_reply"

Using RPC Mode

You can configure an adapter to act as an RPC (remote procedure call) server on behalf of a client. TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) uses the TIBCO Adapter SDK Operations API for RPCServer. Selecting RPC mode creates an object in the repository describing the RPC server that the adapter instance will start.

This section presents the structure of the TIBCO Repository class objects that provide `MOperation` support. Based on these class descriptions, you can create an RPC client to send requests to the adapter in the expected structure.

For convenience, class structure is shown in XML format.

One-way and Two-way Invocation

You can use one-way or two-way messaging in an RPC operation.

- **One-way messaging**

  The BusinessWorks client sends a request containing a message to the adapter server, and no reply is expected from the adapter server. The client does not need to wait for the reply and is free to continue processing the request. The request is not always sent immediately to the server but waits in a queue until the server executes it.

- **Two-way messaging**

  In an RPC operation, two-way messaging is used by default. The BusinessWorks client waits until it gets an answer from the adapter server before continuing processing. If the BusinessWorks client does not receive a reply from the adapter server within a certain period of time, a time-out occurs. The client cannot send a second request until the first request is executed or until a time-out occurs.

Schema Types

Two schema types can be used for the server object created in the repository.

- Standard RPC Operation—the predefined standard request and reply object schema by TIBCO ActiveMatrix Adapter for Database. This schema can be used to describe the input and output of any database request, allowing a Request-Response Service to process database operations on any table or execute any stored procedure.

- Custom RPC Operation—when you want to integrate with third-party applications to describe the actual input and output values of a database operation or a stored procedure. You can use this schema to define your operations, and specify the stored procedures to be executed at design time. These operation definitions are stored in the repository in a way that allows easy integration with other applications.

Standard RPC Operation

The repository contains descriptions of the classes and operations provided for `MOperation` support.

The `SQL_OPS` class describes the operations that the adapter instance, acting as an RPC server, can handle.

The structure of the server object is:

```
<servers>
      <rvCmRpcServer
        name = "agentNamereqreprvcmRPCServer"
        session = "agentNamepubreqreprvcmRvCmSession"
        subject = "ADB.SDK.OPERATION"
        classRef = "SQL_OPS"
      />
</servers>
```

Two types of operations are supported, `SQL_EXECUTE` and `SQL_BATCHEXECUTE`. `SQL_EXECUTE` takes a single SQL statement and processes it. `SQL_BATCHEXECUTE` takes a sequence of SQL statements and processes them. The other classes, such as `SQL_STATEMENT`, `SQL_BIND`, and `SQL_RETURN`, describe metadata for the `input` and `output` parameters to the operations.

- Use `SQL_BATCHEXECUTE` to send one or more SQL statements in a request.
- `SQL_BATCHRETRUN` is the return class.
- If an error occurs while executing a statement, the adapter returns the error immediately without executing the remaining statements.
- On Oracle only, all statements are executed within the same transaction.

On Sybase database, it is not recommended to use `SQL_BATCHEXECUTE` to execute the `ddl` statements. If you have to execute the `ddl` statements, you must set the parameter `ddl in tran` to `true` in the database.

SQL_OPS Class

The structure of the `SQL_OPS` class is:

```
<class
  name = "SQL_OPS">
  <operation
   name = "SQL_EXECUTE"
   returnClass = "SQL_RETURN">
   <parameter name = "STATEMENT"  classRef = "SQL_STATEMENT"
    direction = "in"> </parameter>
  </operation>
  <operation
   name = "SQL_BATCHEXECUTE"
   returnClass = "SQL_BATCHRETURN">
   <parameter name = "STATEMENTS" classRef =
    "sequence[SQL_STATEMENT]"
     direction = "in" />
  </operation>
 </class>
```

SQL_RESULTSET Class

The structure of the `SQL_RESULTSET` class is:

```
<class name = "SQL_RESULTSET">
  <attribute name = "HEADER" class = "sequence[string]">
  </attribute>
  <attribute name = "ROWVALUES" class = "sequence[SQL_ROW]">
  </attribute>
  <attribute name = "OUTBINDS" class = "sequence[SQL_BIND]">
  </attribute>
</class>
```

SQL_ROW Class

The structure of the `SQL_ROW` class is:

```
<class name = "SQL_ROW">
  <attribute name = "ROW" class = "sequence[any]">
  </attribute>
 </class>
```

SQL_RETURN Class

The structure of the `SQL_RETURN` class is:

```
<object name="SQL_RETURN" lastModified="1046487158293" id="202">
  <assoc name="attribute">
    <string name="name" value="STATUS"/>
    <ref name="attributeType" value="/tibco/public/scalar/ae/string"/>
    <string name="isKey" value="false"/>
    <string name="isReadable" value="true"/>
    <string name="isWriteable" value="true"/>
  </assoc>
  <assoc name="attribute">
    <string name="name" value="SQL"/>
    <ref name="attributeType" value="/tibco/public/scalar/ae/string"/>
    <string name="isKey" value="false"/>
    <string name="isReadable" value="true"/>
    <string name="isWriteable" value="true"/>
  </assoc>
  <assoc name="attribute">
    <string name="name" value="ERROR_DESC"/>
    <ref name="attributeType" value="/tibco/public/scalar/ae/string"/>
    <string name="isKey" value="false"/>
    <string name="isReadable" value="true"/>
    <string name="isWriteable" value="true"/>
  </assoc>
  <assoc name="attribute">
    <string name="name" value="CLOSURE"/>
    <ref name="attributeType" value="/tibco/public/scalar/ae/any"/>
    <string name="isKey" value="false"/>
    <string name="isReadable" value="true"/>
    <string name="isWriteable" value="true"/>
  </assoc>
  <assoc name="attribute">
    <string name="name" value="RETURNVALUE"/>
    <ref name="attributeType"
value="/tibco/public/class/ae/ADB/adbmetadata/SQL_RETURNVALUE"/>
    <string name="isKey" value="false"/>
    <string name="isReadable" value="true"/>
    <string name="isWriteable" value="true"/>
  </assoc>
    <string name="family" value="ae"/>
    <string name="objectType" value="class"/>
</object>
```

SQL_RETURNVALUE Class

The structure of the SQL_RETURNVALUE class is:

```
<object name="SQL_RETURNVALUE" lastModified="1046487158293" id="204">
  <assoc name="attribute">
    <string name="name" value="OUTBINDS"/>
    <ref name="attributeType"
value="/tibco/public/sequence/ae/class/ae/ADB/adbmetadata/sequence[SQL_BIND]"/>
    <string name="isKey" value="false"/>
    <string name="isReadable" value="true"/>
    <string name="isWriteable" value="true"/>
  </assoc>
  <assoc name="attribute">
    <string name="name" value="RESULTSETS"/>
    <ref name="attributeType"
value="/tibco/public/sequence/ae/class/ae/ADB/adbmetadata/sequence[SQL_RESULTSET]"/
>
    <string name="isKey" value="false"/>
    <string name="isReadable" value="true"/>
    <string name="isWriteable" value="true"/>
  </assoc>
    <string name="family" value="ae"/>
    <string name="objectType" value="class"/>
</object>
```

SQL_ BATCHRETURN Class

The structure of the `SQL_BATCHRETURN` class is:

```
<class name = "SQL_BATCHRETURN">
  <attribute name = "STATUS" class = "string">
  </attribute>
  <attribute name = "RESULTSETS" class = "sequence[SQL_RESULTSET]">
  </attribute>
  <attribute name = "SQL" class = "string">
  </attribute>
  <attribute name = "ERROR_DESC" class = "string">
  </attribute>
  <attribute name = "CLOSURE" class = "any"
  </attribute>
</class>
```

SQL_BIND Class

The structure of the `SQL_BIND` class is:

```
<class name = "SQL_BIND">
  <attribute name = "POSITION" class = "i4">
  </attribute>
  <attribute name = "TYPE" class = "string">
  </attribute>
  <attribute name = "DATA" class = "any">
  </attribute>
  <attribute name = "NAME" class = "string">
  </attribute>
</class>
```

SQL_STATEMENT Class

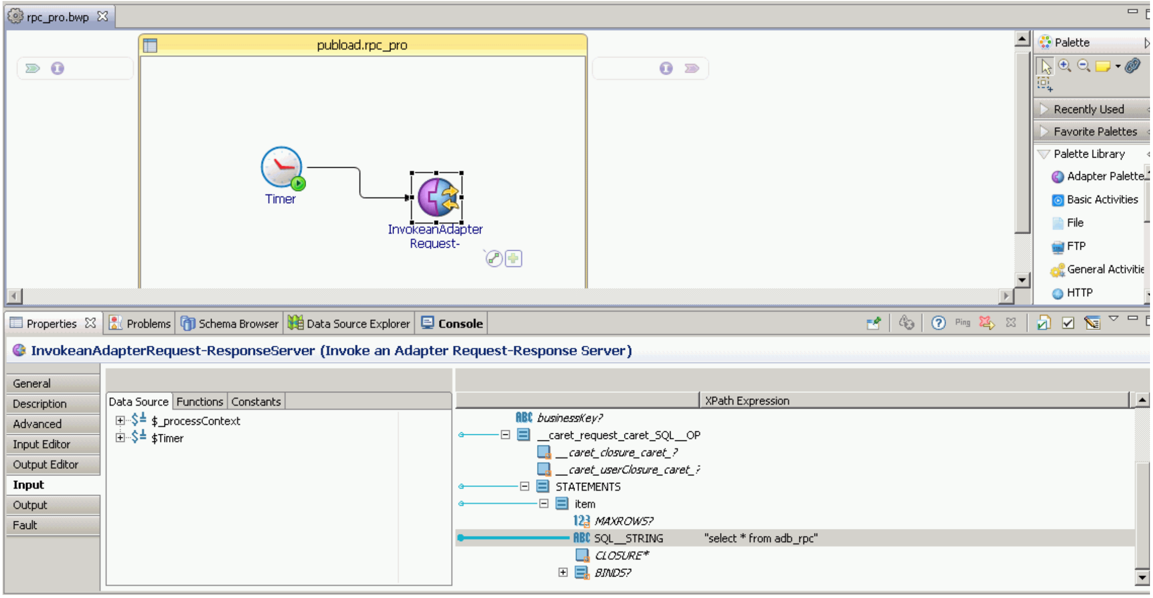
The structure of the `SQL_STATEMENT` class is:

```
<class name = "SQL_STATEMENT">
  <attribute name = "SQL_STRING" class = "string">
  </attribute>
  <attribute name = "BINDS" class = "sequence[SQL_BIND]">
  </attribute>
  <attribute name = "CLOSURE" class = "any"
  </attribute>
  <attribute name = "MAXROWS" class = "i4">
  </attribute>
</class>
```

Figure 6 indicates the input when using BusinessWorks to process the standard RPC operation.

When calling a Request-Response Service, the CLOSURE and DATA elements of anyType AE data type are used in the Adapter activity. You need to manually substitute anyType with a data type that matches the database metadata. See Handling the anyType AE Data Type on page 90 for the detailed information.

Figure 6 Processing the Standard RPC Operation Using BusinessWorks



InvokeanAdapterRequest-ResponseServer (Invoke an Adapter Request-Response Server)

```
businessKey?
__caret_request_caret_SQL__OP
  __caret_closure_caret_?
  __caret_userClosure_caret_?
  STATEMENTS
    item
      MAXROWS?
      SQL__STRING        "select * from adb_rpc"
      CLOSURE*
      BINDS?
```

Custom RPC Operation

When you select the Use Custom Operations check box in the Request-Response Options tab of the Request-Response Service, you can create custom schema for your operations. You can define the call operations to be executed by the RPC server in advance. TIBCO Business Studio automatically generates the input and output schema for the call operation.

The operation takes one input class, REQUEST, which describes the input parameters and other input options for the stored procedure execution. The operation returns a class called REPLY, which describe the output schema of the stored procedure or any error message the agent returns.

Table 51 Request Schema Description

<table>
<tr><th>Input Item</th><th>Data Type or Classname</th><th>Description</th></tr>
<tr><td>INBINDS</td><td>INPUT_BINDS</td><td>The input parameters of the stored procedure.</td></tr>
<tr><td>OPTIONS</td><td>INPUT_OPTIONS</td><td>The input options:
<table>
<tr><th>Option</th><th>Data Type</th><th>Description</th></tr>
<tr><td>MAXROWS</td><td>Integer</td><td>The maximum number of rows to retrieve.</td></tr>
<tr><td>SQL</td><td>String</td><td>The SQL string uses to execute the stored procedure. This string is automatically generated by the palette.</td></tr>
<tr><td>CACHE</td><td>Boolean</td><td>True if users want the agent to cache the statement for performance optimization.</td></tr>
<tr><td>PACKAGE</td><td>String</td><td>Read only. Uses the call operation form to modify the package of the stored procedure.</td></tr>
<tr><td>SCHEMA</td><td>String</td><td>Read only. Uses the call operation form to modify the package of the stored procedure.</td></tr>
</table></td></tr>
<tr><td>CLOSURE</td><td>Any</td><td>Closure argument. The reply returns this closure argument untouched.</td></tr>
</table>

Table 52 Reply Schema Description

Output Item	Data Type or Classname	Description
OUTBINDS	OUTPUT_BINDS	The output parameters of the stored procedure
RESULTSET[1..n]/ RESULTSETS	OUTPUT_ROWS/ SQL_RESULTSET	The result set(s) returned by the stored procedure, see section 'Result Set Support' for more description.

Table 52 Reply Schema Description (Cont'd)

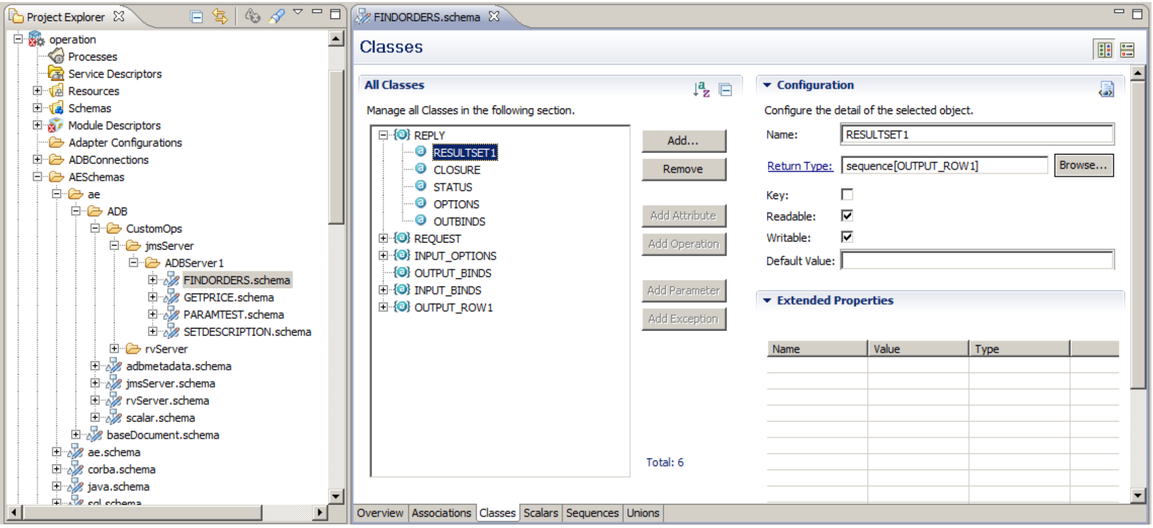
Output Item	Data Type or Classname	Description
`STATUS`	String	`SUCCESS` if the stored procedure is executed successfully. `FAILURE` if there is an error. Error details are stored in the `OPTIONS` class.
`OPTIONS`	`CUSTOM_OP_OUTPUT_OPTION`	Contains the error description and the SQL statement if an error occurred.
`CLOSURE`	Any	Closure argument obtained from the request.

Resultset Support

TIBCO ActiveMatrix Adapter for Database supports multiple resultsets returned for a stored procedure. The class generated for the resultset output depends on the schema information provided by the JDBC driver.

- For drivers that return valid information of a resultset schema, TIBCO ActiveMatrix Adapter for Database creates an output class, named `OUTPUT_ROW[1..n]`, for each result set. This is shown in Figure 7.

Figure 7 Resultset Schema for Drivers That Return Valid Information



Classes

All Classes

Manage all Classes in the following section.

Total: 6

Configuration

Configure the detail of the selected object.

Name: RESULTSET1

Return Type: sequence[OUTPUT_ROW1]

Key: □

Readable: ☑

Writable: ☑

Default Value:

Extended Properties

Name	Value	Type

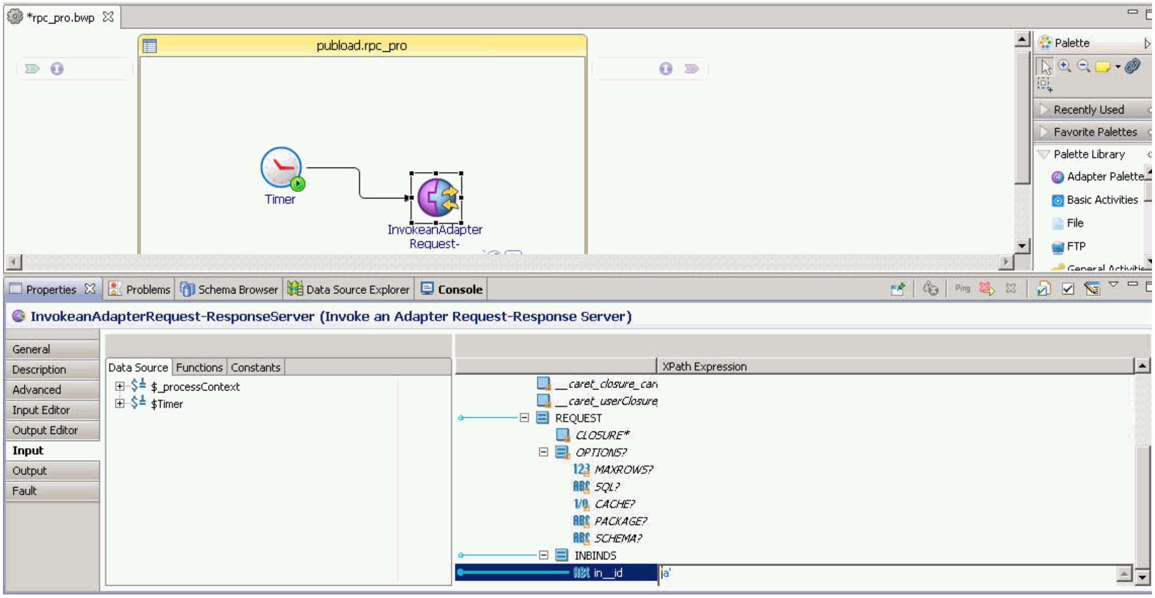
- If a driver does not return valid information for the resultset schema, TIBCO ActiveMatrix Adapter for Database uses the generic class `SQL_RESULTSETS` as the output schema, as described in Standard RPC Operation on page 174.

DataDirect JDBC 5.1.0 drivers only provide valid resultset schema information for Oracle Server version 11.1.0 or higher.

Figure 8 indicates the input when using BusinessWorks to process the custom RPC operation.

When calling a Request-Response Service, the CLOSURE and DATA elements of anyType AE data type are used in the Adapter activity. You need to manually substitute anyType with a data type that matches the database metadata. See Handling the anyType AE Data Type on page 90 for the detailed information.

Figure 8 Input of the Custom RPC Operation Using BusinessWorks



Using the REF Data Type with Oracle Databases

The adapter supports the use of the REF data type (cursor) as an OUT parameter only in an Oracle stored procedure. The adapter returns a result in the same way as it does a result set from a SELECT query. The exact usage depends on the driver used. When using DataDirect Connect® (Merant) drivers, the procedure is called by leaving the REF cursor parameter out of the procedure's call list. For example, the procedure `ORDER_DEMO.FindOrders` with an input price of $1.00 can be called through the request program with the following request:

```
{stmt={sql="{call ORDER_DEMO.FindOrders(?)}" bind={position=1
type="IN" data=1}}}
```

The REF cursor's data will be returned in the results section of the reply.

Considering the following PL/SQL procedure:

```
-- Create a REF cursor procedure
CREATE OR REPLACE PACKAGE ORDER_DEMO AS
   TYPE OrderCurTyp IS REF CURSOR RETURN REPLYTEST%ROWTYPE;
   PROCEDURE FindOrders (price IN NUMBER, order_cv IN OUT
OrderCurTyp);
END ORDER_DEMO;

CREATE OR REPLACE PACKAGE BODY ORDER_DEMO AS
   PROCEDURE FindOrders (price IN NUMBER, order_cv IN OUT
OrderCurTyp) IS
   BEGIN
      OPEN order_cv FOR SELECT * FROM REPLYTEST where ID > price;
   END FindOrders;
END ORDER_DEMO;
```

Using Quotes in a Request with Microsoft SQL Server Databases

When constructing a request in your application that will be processed by the adapter with a Microsoft SQL Server database, you must take care when using quotes.

For example, the following procedure is part of a request from an application. Double quotation marks are used, which is incorrect. An error will be returned.

```
select @qry = "Update " + @tablename + " set ORDER_DESCRIPTION =
'UPDATE TEST'" + ", ORDER_PRICE = 10109.25"
```

The following procedure is the same as above, but uses single quotes. It will be correctly processed.

```
select @qry = 'Update ' + @tablename + ' set ORDER_DESCRIPTION =
''UPDATE TEST'', ORDER_PRICE = 10109.25'
```

See *Delimited Identifiers* in your Microsoft SQL Server documentation for details.

The SQL statement should contain only ASCII characters.

If your SQL statements contain non-ASCII characters, convert them into a stored procedure and invoke the procedure using custom RPC operations.

Chapter 10 Performance Tuning

Besides tuning hardware and memory settings, you can also tune the configuration settings of adapter services to optimize performance.

Topics

Multithreading

Using multi threads, an application can simultaneously process multiple, independent events. TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) supports multithreading to efficiently handle requests and increase the response speed of the system.

When a group message comes, no matter how many records in the group, they will be processed by a same thread as a whole.

Multithreading in Publication Services

Each Publication Service can be configured with multiple threads that connects to the database separately, and processes requests in parallel. You can set the number of threads for the service with the **Number of Publication Service Threads** field in the Publication Options tab.

Load balancing and multithreading cannot co-exist in the Publication Service.

Multithreading in Subscription Service

By using multithreading, the adapter spreads the subscription message load by using multithreading and avoids blocking the processing on a specific subscription table.

- TIBCO Hawk statistics display all Subscription Service thread information and operation count.
- Debug messages provide the Subscription Service thread name in order to distinguish which message is from which Subscription Service thread.
- If a Subscription Service thread encounters fatal errors that the adapter cannot handle, the adapter terminates. The examples of the fatal errors include:
  - Fail to roll back a transaction,
  - Fail to insert received data into the destination table, but the exception table is not configured,
  - Fail to insert error data into the exception table, but the opaque exception table is not configured,
  - Fail to insert error data into opaque exception table,
  - and so on.

Default Session Without Multithreading

In the default session, the Subscription Service of the same transport type without using multithreading shares one session to receive the messages and updates the relevant tables in its associated database.

To use a Subscription Service in single threaded mode using the default session, you need to clear the **Use Separate Session** check box in the Subscription Options tab of the service.

Separate Session With Multithreading

You can configure a Subscription Service to use separate sessions that permit the service to operate using multiple threads.

1. Select the **Use Separate Session** check box in the Subscription Options tab of the service. The useSerial check box and the Number of Subscription Service Threads field are displayed.
2. Select or clear the **useSerial** check box depending on whether you need the message to be handled in order.

   To ensure the proper functioning of the useSerial option, you need to pay attention to the following points:

- — The destination table should have a primary key.
- — If the destination table has no primary key column, you need to manually specify one or more columns by selecting the **User Key** check box. When processing the application requests, the message is handled in order.
- — If the destination table has no primary key column, and the **User Key** check box is cleared, then it is recommended to clear the **useSerial** check box.

3. Specify the number of threads.

When you need to use multiple threads by configuring a separate session, the transport session name must end with one or more digits from [0-9].

Parent-Child Table versus Single Table

As mentioned in Chapter 8, Configuring Subscription Services, Subscription Services can be of two types: those with a single table and those with multiple tables.

- For Subscription Services with a single table, because only one table is concerned, multithreading helps improve performance.
- For Subscription Services with multiple tables, the tables form a tree. That is, a table is either the parent of some of the rest tables or a child of another table, or both. Performance improvement gained by multithreading may be minimal. This is because a thread accesses tables in exclusive mode for data integrity. So a table currently being processed by a thread is inaccessible to other threads. This table lock must be released by another thread before it can be used, negating any multi-threading advantages on the adapter side.

Other factors also affect the performance of Subscription Service with multiple tables, such as the number of the tables involved, the number of table columns involved, and the level of the hierarchies of the table tree.

Multithreading in Request-Response Service

You can improve the adapter Request-Response Service performance by specifying the number of threads that are responsible for processing application requests. Each Request-Response Service thread is dedicated to listen on an agreed request subject.

You can set multithreading at the adapter configuration level or the adapter service level.

- To set multithreading at the adapter configuration level: use the **Number of Request-Response Service Default Session Threads** field in the **Advanced** tab of the adapter configuration.

- To set multithreading at the adapter service level: use the Request-Response Options tab of the service configuration:

  a. Select the **Use Separate Session** check box.

  b. Specify the **Number of Request-Response Service Threads** field.

Default Session Without Multithreading

When setting the multithreading option at the adapter configuration level and using the default setting at the adapter service level, the Request-Response Service with the same transport type shares one session to process the request and returns the results in response to the client no matter how many services in this adapter configuration.

Separate Session With Multithreading

When setting the number of Request-Response Service threads at the adapter service level, each Request-Response Service can hold a separate session with multiple threads that process the request and returns the results in a response to the client separately.

If you set multithreading at both adapter configuration level and adapter service level, the Request-Response Service will take the thread number at the adapter service level during the process.

When you need to use multiple threads by configuring a separate session, the transport session name must end with one or more digits from [0-9].

Load Balancing

In order to improve the performance in high load scenario, TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) supports the configuration of load balancing to distribute workload across services.

Load Balancing in Publication Services

The following example describes how to configure the load balance in design time for Publication Services. Publication Services with the same Mutex name specified are in the same load balance group.

When working with the load balancing in Publication Services, you need to pay attention to the following points:

- When creating two or more Publication Services, you must load same source tables to all Publication Services in each publication adapter configuration.
- The automatically generated publishing tables must be the same for all Publication Services in each Publication instance you created. To ensure the publishing tables are the same, if you select the **Write to Database on Save** check box in one adapter configuration, you must clear this check box in other adapter configurations.
- When you specify the **Mutex Name** field the Publication Options tab for all Publication Services in each Publication instance you created, you must make sure the mutex name are all the same.
- When you specify the message subject or destination name in the Configuration tab for all Publication Services in each Publication instance you created, you must make sure the message subject or destination names are all the same.

To finish this example, you need to perform the following tasks:

- Task A, Create the Adapter for Database Connection, page 190
- Task B, Create the Publication Adapter Configuration, page 191
- Task C, Add the Publication Service, page 191
- Task D, Configure the Load Balancing, page 192

Task A Create the Adapter for Database Connection

This example uses an Oracle database connection. To create the connection, complete the following steps:

1. Start TIBCO Business Studio and create a project, for example, **ADADB_loadbalancing**.
2. From the main menu, click **File > New > Other**.
3. In the **New** dialog, specify the search filter **adapter**. Then select **TIBCO Adapters > Adapter for Database > Adapter for Database Connection** and click **Next**.
4. In the **Adapter for Database Connection** dialog, specify **Oracle6694** in the Connection Resource field and other connection fields according to the details in Add a Database Connection Resource on page 38.

   Click **Test Connection** to verify the connection parameters are correct and click **OK**. Then click **Finish** to create the connection resource.

Task B Create the Publication Adapter Configuration

To create the Publication Service, complete the following steps:

1. Click File > New > Other.
2. In the New dialog, specify the search filter **adapter**. Then select **TIBCO Adapters > Adapter for Database > Adapter for Database Configuration** and click **Next**.
3. In the Adapter for Database Configuration wizard, select the **ADADB_loadbalancing** project folder. Specify the file name for the adapter configuration. Click **Finish**.
4. In the Configuration tab:

   — Select the database type you are using, such as **`Oracle`**, from the Vendor list.

   — Select the **Write To Database On Save** check box. This default option writes configuration information to the database when you save the project in TIBCO Business Studio.

   — Click the **...** browse button next to the Connection Reference field. From the Adapter for Database Connection Configuration dialog, select **`Oracle6694.sharedadbc`**. Click **OK**.
5. Repeat step 1 to step 4 to creating another Publication adapter configuration, for example, **`test_pub2`**. Note that you need to clear the Write to Database on Save check box.

Task C Add the Publication Service

To add the Publication Service, complete the following steps:

1. In the Project Explorer view, double click `test_pub1.adadbmodel`.
2. Click the Adapter Services tab in the adapter configuration editor.
3. Click **Add** in the All Adapter Services panel.
4. In the Create Service Wizard, select **Publication** from the **Service Type** list. Click **Next**.
5. In the Schema Type page, pick a table for the service:
   a. Click the **...** browse button to open the ADB Business Object Schema Picker dialog.
   b. Click the Remote Business Object tab.
   c. Expand the **Oracle6694** node.
   d. Right-click the **table** node, and from the pop-up menu click **Fetch Table**.
   e. In the Table name Pattern dialog, specify the search criteria, and click **OK**.
   f. In the Table Download Dialog, select the table from the list. Click **OK**.
   g. Select the fetched table in the Remote Business Object tab. Click **OK**.
6. (Optional) If you want to select a transport session, click **Next**. In the Transport Session page, you can select a different transport session.
7. Click **Finish**.
8. Repeat step 1 to step 7 to add the Publication Service to `test_pub2` adapter configuration. Note that for step 5, click the Local Business Object tab and select the table you just fetched.

Task D Configure the Load Balancing

To configure the load balancing, complete the following steps:

1. Click the **test_pub1.adadbmodel** tab in the adapter configuration editor.
2. Click the Adapter Services tab.
3. Select **ADBPublisher** from the All Adapter Services panel.
4. In the Configuration tab, specify the name in the Destination field. For example, `test_loadbalancing_pub`.
5. In the **Publication Options** tab, select the **Use Polling Batch Size** check box, and set a value in the Polling Batch Size (Maximum Rows) field.
6. Select the **Enable Load Balancing** check box, the Mutex Name field is displayed.

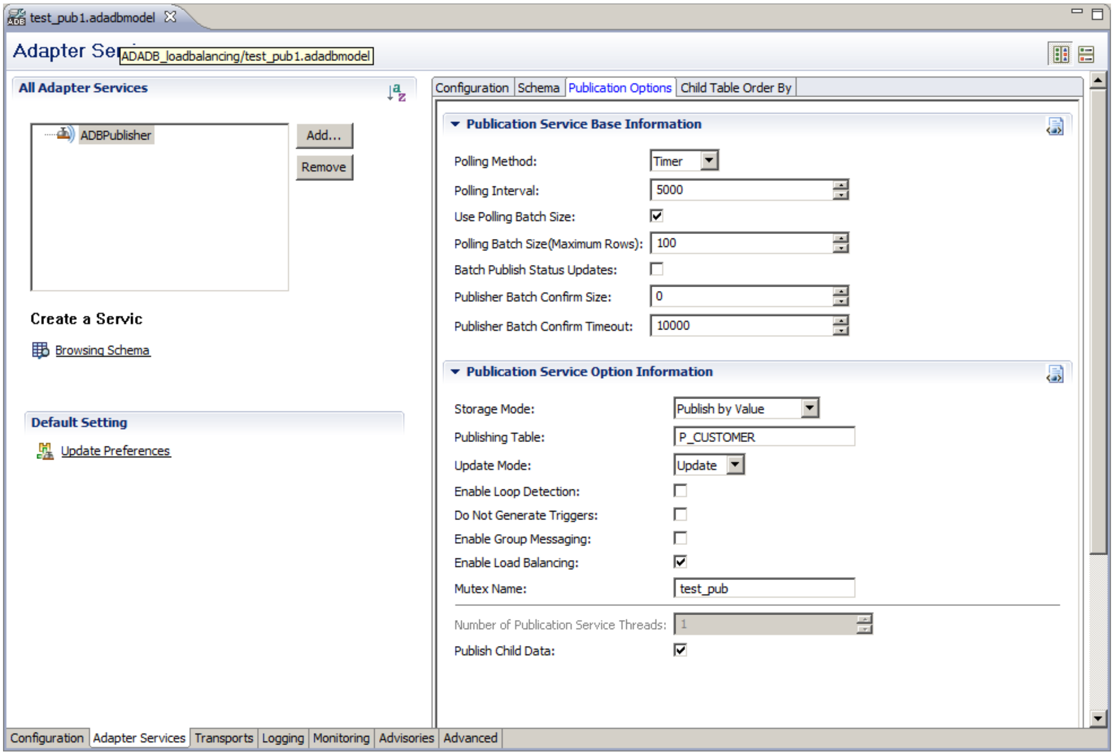
7. Select one of the following two ways to create the mutex table:

   — Enter a name in the Mutex Name field, and the Publication Service will create a mutex table. For example, **`test_pub`**.

   — Users can define a mutex table by themselves, then enter the mutex table name into the Mutex Name field. For example, users can use the following SQL statement to create the mutex table in the load balancing mode:

   ```
   CREATE TABLE test_pub (COL1 INT)
   ```

Figure 9 Configuring Load Balancing in the Publisher Options Tab



Publication Service Base Information

Polling Method: Timer

Polling Interval: 5000

Use Polling Batch Size: ☑

Polling Batch Size(Maximum Rows): 100

Batch Publish Status Updates: ☐

Publisher Batch Confirm Size: 0

Publisher Batch Confirm Timeout: 10000

Publication Service Option Information

Storage Mode: Publish by Value

Publishing Table: P_CUSTOMER

Update Mode: Update

Enable Loop Detection: ☐

Do Not Generate Triggers: ☐

Enable Group Messaging: ☐

Enable Load Balancing: ☑

Mutex Name: test_pub

Number of Publication Service Threads: 1

Publish Child Data: ☑

8. Save the project.

9. Repeat step 1 to step 8 to configure load balancing in Publication Service **test_pub2**.

At runtime, the Publication Services of the same load balancing group creates a mutex table that was configured, and polls the same publication table and distributes loads. The service synchronizes by locking the mutex table to obtain and update the sequence information.

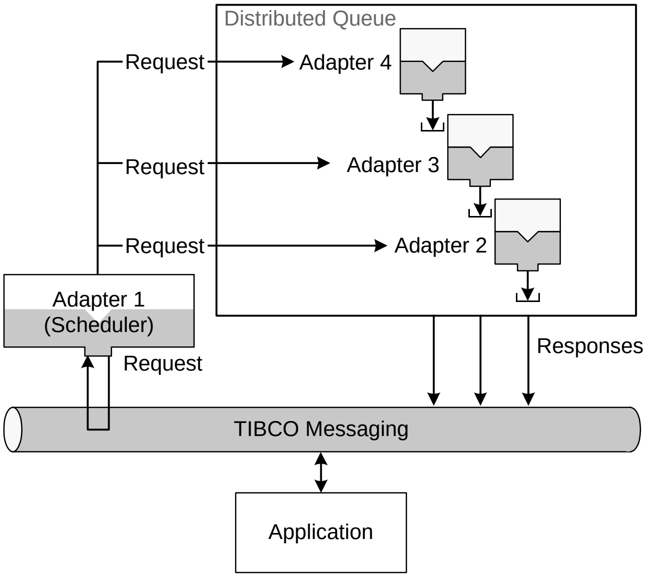
- When an adapter configuration contains a Publication Service with load balancing enabled, you must select the Use Polling Batch Size check box, and set a value in the Polling Batch Size (Maximum Rows) field.
- For Sybase, the publisher load balancing feature requires the `ddl in tran` option be set to true:

  `sp_dboption` *database_name*, `"ddl in tran", true`

Load Balancing in Subscription Services and Request-Response Services

TIBCO Rendezvous Distributed Queuing or TIBCO JMS Queue can be used to achieve load balancing across adapter instances for Subscription and Request-Response Services. Figure 10 is an example for using TIBCO Rendezvous Distributed Queueing for load balancing. It shows three adapter configurations, each connected to a database server (not shown) that contains replicated data.

Figure 10 Load Balancing in Request-Response Services



The adapter configurations have been set up to use distributed queueing with one acting as the scheduler. Only one of the three adapter configurations will receive an incoming request from an application. The scheduler will assign certified requests to the least loaded member of the queue.

A queue member’s load is determined by the number of pending processes that are waiting to be processed by the member’s database Request-Response thread. After the request is processed, the processing configuration sends a response back to the requesting application.

Configure an Adapter Configuration to use a JMS Queue Transport Session for Load Balancing

When working with the load balancing in Subscription or Request-Response Service, you need to pay attention to the following:

- When creating two or more Subscription Service or Request-Response Service adapter configurations, and if you select the Write to Database on Save check box in one adapter configuration, then you cannot select this check box in other adapter configurations.
- When creating two or more Subscription Services, you must load same source tables to all Subscription Services you created.
- When you specify the message subject or destination name in the Configuration tab for all Subscription Services you created, you must make sure the message subject or destination names are all the same.
- When you specify the message subject or destination name in the Configuration tab for all Request-Response Services you created, you must make sure the message subject or destination names are all the same.
- To avoid confusion, do not use multithreading and clear the Use Separate Session check box in the Subscription Options tab.

This example introduces the configuration of Subscription Service adapter configurations using JMS Queue transport session.

- Task A, Create the Adapter for Database Connection, page 190
- Task B, Create the Publication Adapter Configuration, page 191
- Task C, Add the Publication Service, page 191
- Task D, Configure the Load Balancing, page 192

Task A Create the Adapter for Database Connection

This example uses an Oracle database connection. To create the connection, complete the following steps:

1. Start TIBCO Business Studio and create a project, for example, **ADADB_loadbalancing**.
2. From the main menu, click **File > New > Other**.
3. In the **New** dialog, specify the search filter **adapter**. Then select **TIBCO Adapters > Adapter for Database > Adapter for Database Connection** and click **Next**.
4. In the **Adapter for Database Connection** dialog, specify **Oracle6694** in the Connection Resource field and other connection fields according to the details in Add a Database Connection Resource on page 38.

   Click **Test Connection** to verify the connection parameters are correct and click **OK**. Then click **Finish** to create the connection resource.

Task B Create the Subscription Adapter Configuration

To create the Publication Service, complete the following steps:

1. Click File > New > Other.
2. In the New dialog, specify the search filter **adapter**. Then select **TIBCO Adapters > Adapter for Database > Adapter for Database Configuration** and click **Next**.
3. In the Adapter for Database Configuration wizard, select the **ADADB_loadbalancing** project folder. Specify the file name `test_sub1` for the adapter configuration. Click **Finish**.
4. In the Configuration tab:
   - Select the database type you are using, such as `Oracle`, from the Vendor list.
   - Select the **Write To Database On Save** check box. This default option writes configuration information to the database when you save the project in TIBCO Business Studio.
   - Click the **...** browse button next to the Connection Reference field. From the Adapter for Database Connection Configuration dialog, select `Oracle6694.sharedadbc`. Click **OK**.
5. Repeat step 1 to step 4 to creating another Publication adapter configuration, for example, `test_sub2`. Note that you need to clear the Write to Database on Save check box.

Task C Add the Subscription Service

To add the Subscription Service, complete the following steps:

1. In the Project Explorer view, double click `test_sub1.adadbmodel`.
2. Click the Adapter Services tab in the adapter configuration editor.
3. Click **Add** in the All Adapter Services panel.
4. In the Create Service Wizard, select **Subscription** from the **Service Type** list. Click **Next**.
5. In the Schema Type page, pick a table for the service:
   a. Click the **...** browse button to open the ADB Business Object Schema Picker dialog.
   b. Click the Remote Business Object tab.
   c. Expand the **Oracle6694** node.
   d. Right-click the **table** node, and from the pop-up menu click **Fetch Table**.
   e. In the Table name Pattern dialog, specify the search criteria, and click **OK**.
   f. In the Table Download Dialog, select the table from the list. Click **OK**.
   g. Select the fetched table in the Remote Business Object tab. Click **OK**.
   h. Click **Next**.
6. In the Transport Session page, select a JMSQueue transport session:
   a. Click the **...** browse button to open the Model Select dialog.
   b. Click **New** to open the New Transport dialog.
   c. Specify a name in the **Transport Name** field.
   d. Select **JMS** from the **Transport Type** list.
   e. Select **JMS Queue Connection**.
   f. Click **Finish**.
   g. Select the created JMS Queue transport session.
   h. Click **OK**.
7. Click **Finish**.
8. Repeat step 1 to step 7 to add the Subscription Service to `test_sub2` adapter configuration. Note that for step 5, click the Local Business Object tab and select the table you just fetched.

Task D Configure the Load Balancing

To configure the load balancing, make sure the destination name in the Configuration tab is the same for all Subscription Services you created.

Batch Processing

The adapter supports batch processing for Publication Services and Subscription Services.

Batch Processing in Publication Services

A Publication Service publishes inserted data from the publishing table, which copies the data from the source table, and update message status to the publishing table. You can configure the service to publish data and update message status in batches of specified size. If you are using certified message delivery, you can also configure the service to send confirmation in batches.

You can perform all the configuration using the Publication Options tab of an adapter service.

Having multiple instances of the adapter configured against the same table for publication is not recommended. Such a configuration may produce multiple publications of the same message or result in a database deadlock occurrence.

To Publish Data and Update Message Status in Batches

Select the **Use Polling Batch Size** check box. The following fields are displayed:

- **Polling Batch Size (Maximum Rows)**

  This is the maximum number of messages that are picked up per polling interval. The adapter returns to the event loop when it is finished sending those messages. Using this option helps process events in an efficient manner. For example, when polling a large number of rows, the adapter works best if a fixed number of rows is specified in this field.

  The default value is 0, which indicates that all new rows should be fetched.

When you use Polling Batch Size and Enable Group Messaging together in the Publication Service, it is recommended that you set the value of Polling Batch Size equal or larger than the value of Group Size to increase the performance. If the Polling Batch Size is set smaller than the Group Size, the group cannot be fully filled.

- **Batch Publish Status Updates**

  Select this check box to optimize publishing performance by batching message status updates to the publishing table. If an adapter stops before a batch update is performed, the status column is not updated. As a result, duplicate messages may be published when the adapter is restarted.

Do not use this option when messages are published using a parameterized subject name.

To Send Confirmation in Batches

If you are using TIBCO Rendezvous certified message delivery, to send confirmation in batches, configure the following settings:

- **Publisher Batch Confirm Size**

  This number indicates how many message status need to be updated in a single batch. Entering a value in this field optimizes performance.

- **Publisher Batch Confirm Timeout**

  This is the number of milliseconds to wait before updating the status column. After this interval, an update is performed even if the batch confirm size value is not reached. The default value is `10000` (10 seconds). A value of `0` means that no timeout interval is used.

If an adapter stops before a batch update is performed, the status column is not updated. As a result, messages that were successfully published might still have a status of P (pending) in the publishing table when the adapter is restarted. In this case, the ledger file contains the correct status information. Smaller batch confirm size or timeout values decrease this risk.

Do not use this feature when messages are published using a parameterized subject name.

Batch Processing in Subscription Services

The adapter supports execution of subscription requests in batches at two operation levels: insert of the incoming messages and commit of the insert, update, and delete operations.

To use this feature, configure the following fields:

- Bulk Insert Size

The adapter when configured for batch execution stores incoming subscription requests in a batch. All incoming messages to insert are stored until this size is reached. Then, a bulk insert operation is performed on the destination table by using a single execution. The batch is emptied when batch size is reached.

This number must be less than or equal to the value in Batch Commit Size. The default value is 1.

- Batch Commit Size

  This is the number of messages to batch before invoking a commit operation. The default value is 0, and the adapter will take the value 1 as the batch commit size when running the Subscription Service.

Batch Commit option is not supported with RVCMQ Quality of Service. The RVCMQ scheduler requires a message to be confirmed before dispatching the next message, which prevents the adapter from operating in batch mode.

- Batch Commit Timeout (milliseconds)

  This is the interval (in milliseconds) to wait before committing the inserted data. After this interval, the inserted data will be committed even if the inserted data size does not reach the Batch Commit Size value. The default value is 10000 milliseconds.

  The batch commit feature does not commit all received messages if the adapter instance terminates before the batch commit value or time out value is met.

  When using RVCMQ, batch commit will timeout after each operation (insert/update/delete) when batch commit size > 1.

When using batch mode, you need to pay more attention to the following:

- Because every network interaction comes with a certain amount of overhead, packaging many incoming requests into a single database interaction minimizes network traffic. However, packaging too many requests together may exceed certain limits imposed by some databases (for example, a Teradata database), concerning the size of a single call.
- Also, having too large a setting introduces a higher latency between the time an adapter receives a request and the time the request is processed by the database.

Using Hints (Publication Services only)

Hints help improve the performance of your queries. When the adapter Publication Service executes a poll operation to fetch data from a table, using hints greatly enhances the query.

Hints are only supported in Oracle and SQL Server databases.

Usage

To use hints, add the following line to the adapter TRA properties file:

adb.*table_name*.poll.hint *hint_value*

where, *table_name* is the name of your table and *hint_value* is the hint.

Examples

- To force an index scan when polling in an Oracle database, set the property as follows:

  ```
  adb.p1.poll.hint /*+INDEX(P1,P1_INDX)*/
  ```

  where `p1` is the publisher table, and `P1_INDX` is the index created on the publishing table.

  The adapter will execute this select query on the publishing table:

  ```
  SELECT /*+INDEX(P1,P1_INDX)*/ * FROM P1 WHERE ID = ?
  ```

- To use `NOLOCK` hint when polling in as SQL Server database, set the property as follows:

  ```
  adb.p1.poll.hint WITH(NOLOCK)
  ```

  where `P1` is the publisher table.

  The adapter will execute this select query on the publishing table, but will not issue shared locks or honor exclusive locks:

  ```
  SELECT * FROM P1 WITH(NOLOCK) WHERE ADB_L_DELIVERY_STATUS ='N'
  ```

- To force an index scan when the adapter fetches records from a child table set the property as follows:

  ```
  adb.C1.poll.hint /*+INDEX(C1,C1_INDEX)*/
  ```

  where `C1` is the child table, and `C1_INDEX` is the index created on the child table.

  The adapter will execute this select query on the child table:

```
SELECT /*+INDEX(C1,C1_INDEX)*/ * FROM C1 WHERE ID = ?
```

Chapter 11 Advanced Topics

This chapter introduces the advanced configuration and deployment topics for TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio).

Topics

Password Handling

You can define the Adapter for Database Connection password value to be a module property. Before starting the adapter, include the runtime password as a client variable in the TRA properties file from TIBCO ActiveMatrix Adapter for Database 7.0. Obfuscate the password using the obfuscate utility installed with TIBCO Runtime Agent. For example, if the password value is defined as `%%myPassword%%`, create a module property named `myPassword` in the module properties section with no value and include the following entry in the TRA file of the adapter:

```
tibco.clientVar.myPassword
```

See Password Handling on page 262 for how to use the obfuscate utility.

Subject and Destination Names

An adapter configuration sends and receives messages using subject names when using the TIBCO Rendezvous transport type, and destination names when using the JMS transport type. Both subject and destination names are structured strings of alphanumeric characters divided into elements by the dot (`.`) character. For example, `Tibco.Tsi` and `News.Sports.Baseball` are valid subject and destination names.

Default subject or destination names are provided in TIBCO Business Studio. You can change these default values to specify a custom subject or destination name. A default subject or destination name, depending on which transport type you choose to configure, is created each time an adapter service is created.

The following are restrictions on subject names and destination names:

- Length of subject and destination names
  - Subject names are limited to a total length of 255 characters (including the dot delimiters). Even though TIBCO Rendezvous has a 255-character subject length limit, it is very unlikely that applications will need to use the full length. If you want to run an adapter configuration against Microsoft SQL Server, you must design your subject space to conform to the length restrictions. There can be at most 100 elements in a subject name. The maximum element length is 127 characters.
  - Destination names are limited to a total length of 249 characters. Even though JMS has a 249-character destination length limit, some of that length is reserved for internal use. The destination space must be designed to conform to the length restrictions. There can be at most 64 elements in a destination name. The maximum element length is 127 characters. Dot separators are not included in element length.
- Wildcard characters

  The asterisk (`*`) and greater than (>) are wildcard characters that can be used when specifying subject names or destination names.

  For example, if messages are published on the `tsi.sales.sports` and `tsi.sales.clothing` subjects and your subscriber adapter listens on the `tsi.sales.*` subject, the configuration will get messages sent on both subjects. This feature also applies to destination names.
- Reserved underscore (_)

  Subject or destination names beginning with underscore (_) are reserved. Application programs cannot send subjects or destinations with an underscore as the first character of the first element, however, `_INBOX` and

_LOCAL are exceptions for subject names, and _INBOX is an exception for destination names. It is also recommended that you do not use tabs, spaces, or any unprintable character.

An underscore can be used elsewhere in subject or destination names. Subject names are case sensitive.

- Empty strings

  Empty strings ("") are not permitted.

- Dot character

  The dot character cannot be incorporated into an element by using an escape sequence.

See *TIBCO Rendezvous Concepts* for a full explanation of subject names and *TIBCO Enterprise Message Service User's Guide* for destination names.

Parameterized Subject or Destination Names

During the usage of TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio), a subject name can be created from one or more columns in a publishing table when using the TIBCO Rendezvous transport type, and a destination name can be created when using the JMS transport type. A subject or destination created this way is known as a parameterized subject or destination. For example, you can use TIBCO Business Studio to define a publication subject as MYSUBJECT.$COLUMN1.$COLUMN2, where COLUMN1 and COLUMN2 are names of columns in the publishing table. For example: TIBCO.ORDER.$ORDER_ID.$ORDER_DESCRIPTION.

On publication, a subject or destination is created from the contents of those columns. This allows receiving applications to filter publications based on the values of certain fields. Parameterized subject or destination names are supported with certified or reliable delivery.

Parameterized Subject and Destination Name Restrictions

The following are restrictions on parameterized subject names and destination names:

- Length of parameterized subject and destination names
  - A TIBCO Rendezvous parameterized subject name can be at most 255 characters in length, with each component no more than 127 characters.
  - A JMS parameterized destination name can be at most 249 characters in length, with each component no more than 127 characters.

In the previous example, if the ORDER_DESCRIPTION column contains a description value that is longer than 127 characters, the parameter name $ORDER_DESCRIPTION is used as the subject or destination in place of the description value.

- Wildcard characters

  The asterisk (*) or greater than (>) characters, or empty strings should not be used as attribute values (VARCHAR, CHAR) in a column that is part of a parameterized subject or destination. Doing so makes the subject or destination invalid.

  For example, if the TIBCO.ORDER.$ORDER_ID.$ORDER_DESCRIPTION parameterized subject or destination name is defined and the following values are inserted into ORDER_TABLE:

  ```
  insert into ORDER_TABLE values(111,'*',88.81);
  ```

  and the following values are inserted into the ORDER_DESCRIPTION table:

  ```
  insert into ORDER_DESCRIPTION TABLE values(112,'>',99.91);
  ```

  the publisher would return an error.

- Data type restrictions

  — Do not use columns that contain Date types as part of a parameterized subject or destination. Date values are likely to contain characters such as dashes and spaces that can cause problems when used as part of a subject or destination.

  — Do not use columns that contain binary types as part of a parameterized subject or destination.

  — Do not insert Float-type data as an attribute value in a column that is part of a parameterized subject or destination. Doing so makes the subject or destination invalid.

- Column name

  Each element cannot represent more than one column. For example, MYSUBJECT.$COLUMN1$COLUMN2 is an invalid parameterized subject name.

- Group messaging

  The group messaging feature is not supported with parameterized subjects or destinations. An error message will display and the adapter will automatically disable the group messaging feature for the publisher. For example, if the TIBCO Rendezvous transport type is used, the error message will be:

  ```
  Cannot use parameterized subject with group Publishing. Disable
  group publishing.
  ```

- Pre-registration

  Pre-registration in the adapter cannot work when a parameterized subject or destination is used.

Preregistering a Certified Subscriber

You can add one or more certified subscribers, or listeners, for a publisher adapter that sends messages on subjects using the TIBCO Rendezvous Certified quality of service. The name of the subscriber is added to the preregistered list of the publisher adapter. The subscriber can be a TIBCO ActiveMatrix Adapter for Database subscriber, or any other TIBCO Rendezvous subscriber.

A subscriber adapter on the preregister list is certified to receive all messages sent on the specified subject using the RVCM quality of service. If a subscriber is not preregistered, it may miss one or more of the initial messages sent by the publisher adapter.

- Wildcard TIBCO Rendezvous subject names are not supported for preregistered listeners.
- Preregistration does not work when parameterized subjects are used.
- When using a Publication Service or TIBCO BusinessWorks process to send messages to a Subscription Service, TIBCO recommends that you configure the pre-registered listeners for the TIBCO Rendezvous Certified Publisher endpoint, to avoid losing data.

To pre-register a certified subscriber, follow these steps:

1. In the adapter configuration editor for the publisher adapter, click the Transports tab.
2. Select **adb*adapter_instance_name*rvcmRvCmSession** from the All Adapter Transports panel.
3. Click its Publisher endpoint. The Configuration panel for this endpoint is displayed.
4. Verify the name of the certified subscriber in the **Pre-registered Listeners** field: `%%Domain%%.%%Deployment%%.%%`*AppName*`%%.%%`*InstanceId*`%%.rr.CM`.

- This name must be the subscriber CM name.
- Variable *AppName* and *InstanceId* are the same as the current adapter name.

5. Save the adapter configuration.

Changing the Location of the Ledger File

The ledger file provides temporary storage for published messages that are sent by a publisher adapter using the TIBCO Rendezvous certified messaging quality of service. Each message is held in the ledger until an acknowledgement is received from the subscriber that the message has been consumed. If there are many subscribers, the ledger file can grow to a substantial size. In this case, it is recommended to store the ledger file in a subdirectory under the default directory, which is *TIB_ADADB_HOME*, rather than the default directory itself.

You can change the directory to store the ledger file in TIBCO Business Studio or modify the TRA file.

- To change the directory of the ledger file in TIBCO Business Studio, perform the following steps:

  a. Select the **adb*adapterInstanceName*rvcmRvCmSession** transport session in the **Transports** tab of the adapter configuration.

  b. Specify the **Ledger File** field in the **Rendezvous Options** panel of the tab.

- To modify the TRA file, perform the following steps:

  a. Change directory to the *TIB_ADADB_HOME*`\bin` directory, and then open the adbagent.tra file with the text editor.

  b. Remove the pound symbol (#) at the beginning of the ledger directory command line. For example:

  ```
  c. tibco.clientVar.DirLedger c:/tibco/adapter/adadb/7.0/ledger
  ```

  d. Save and close the TRA file. When restarting the adapter instance, it writes the ledger file to the new location.

Sending dateTime Data to an Adapter Configuration

When TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) publishes messages including `datetime` data, the wire format is handled implicitly since both publisher and subscriber use the same format.

In many scenarios, however, a separate product acts as the publisher, which means that you must explicitly specify the wire format. For example, an adapter configuration can subscribe to messages published by another TIBCO product, such as TIBCO ActiveMatrix BusinessWorks, after some transformation has been performed.

Messages containing `datetime` data sent to the adapter must have that data in a string format. Formatting dates as strings allows the adapter to handle date values outside the range allowed by `RVMSG_DATETIME`, January 1, 1970 to January 1, 2034.

The default format used by the `adbDateTime` metadata class is described in this section. The adapter TRA properties also include a few date and time patterns that you can specify to use.

adbDateTime Object

Do not change any `adbDateTime` object information in this screen. Use this information to determine how to format the third-party subscriber date value.

To view the structure of the `adbDateTime` class, go to the Project panel, and expand **AESchemas > ae > ADB > adbmetadata > Classes > adbDateTime > dateTime**

The input is a string value with one of the following formats:

```
yyyy-mm-dd
yyyy-mm-dd hh:mm:ss.xxx
```

where xxx represents the millisecond value.

Timezone values are not supported.

The adapter returns an error when the input data contains timezone information. The timezone information can be parsed using the TIBCO BusinessWorks XPath functionality before passing it to the adapter.

Specifying Date and Time Patterns

You can use the following TRA properties to specify the date, time, and timestamp format patterns:

- **`adb.datePattern`**

  This is the date format pattern for `java.text.SimpleDateFormat`. The default format is *yyyy-MM-dd*.

- **`adb.timePattern`**

  This is the time format pattern for `java.text.SimpleDateFormat`. The default format is *HH*:*mm*:*ss*.

- **`adb.timestampPattern`**

  The standard timestamp format pattern for `java.text.SimpleDateFormat`. The default format is *yyyy-MM-dd HH*:*mm*:*ss*.*S*.

Using the User Callout Java Library

The user callout Java library allows the transformation of a message that the adapter publishes into a structure that you want to publish. The Java library can be customized to apply limited transformations to outgoing and incoming messages.

If a message is modified before it is sent by a publisher adapter, all subscribers will receive the modified message. If a message is modified before it is received by a subscriber adapter, only that adapter will get the modified message.

For example, at the publisher adapter, a message can be modified to have a field added, or publish an empty message if a certain criteria is met. At the subscriber adapter, a message can be modified to be written to the database or not, based on a filter, or have the subject name inserted into the table.

Before a message is sent or received, the adapter checks the callout library. If the callout library has not been modified, the message is passed back to the adapter unchanged. If the callout library has been modified, the message is passed back to the adapter with the modification. After receiving a modified message from the callout library, the adapter configuration sends or consumes the message like any other message.

Building the Callout Library

The callout library is built with TIBCO Adapter SDK and TIBCO Rendezvous software. For compiler requirements and instructions, see the TIBCO Adapter for SDK guide.

To build the callout library, you need to complete the following tasks:

- Adding a User Library in a Project, page 213
- Adding a Java Package and Class, page 217
- Using the alterMsgPub and alterMsgSub Functions, page 218
- Using the adbPreCommit Function, page 220
- Exporting the JAR File, page 221
- Replacing the Original JAR File, page 223

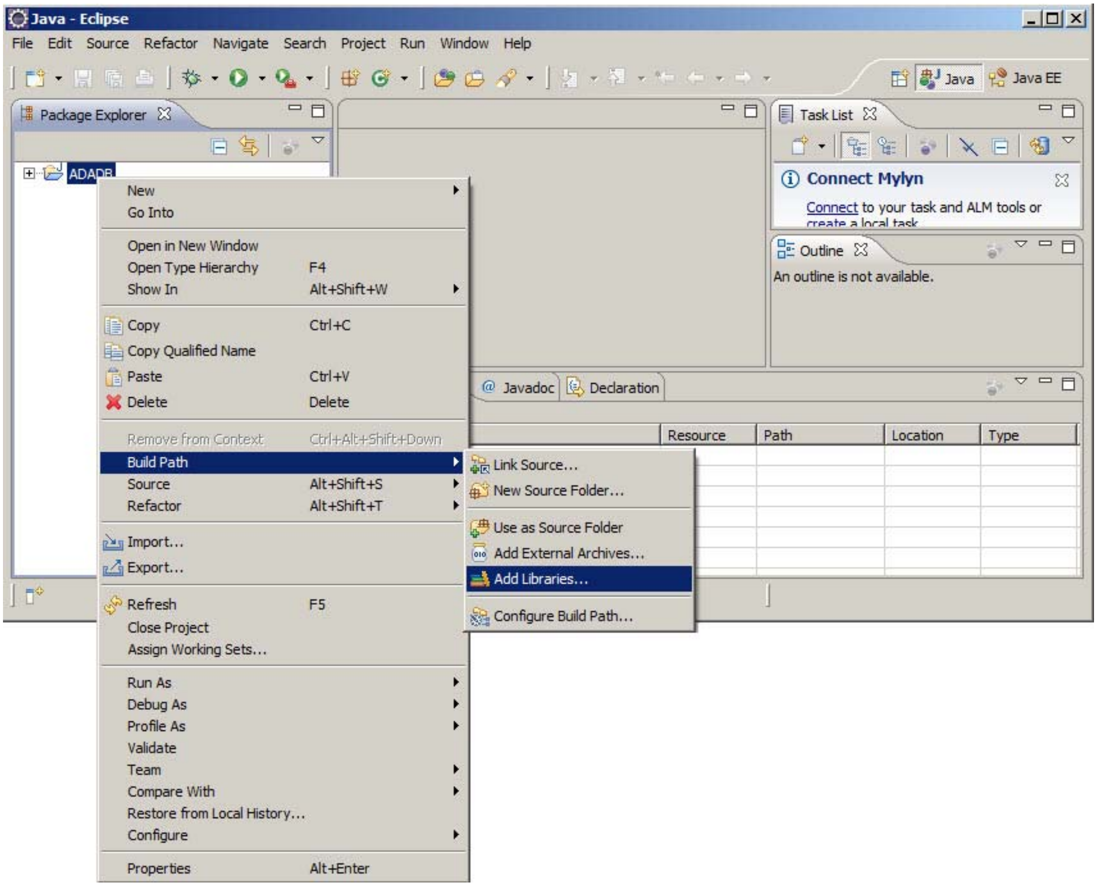
Adding a User Library in a Project

To add user libraries in a project, follow these steps:

1. Using Eclipse, create a new workspace and insert a new project into the workspace.

2. Right-click the new project, then select **Build Path > Add libraries...** from the menu that is displayed. The Add Library dialog is displayed. See Figure 11.

Figure 11 Adding Library in Eclipse

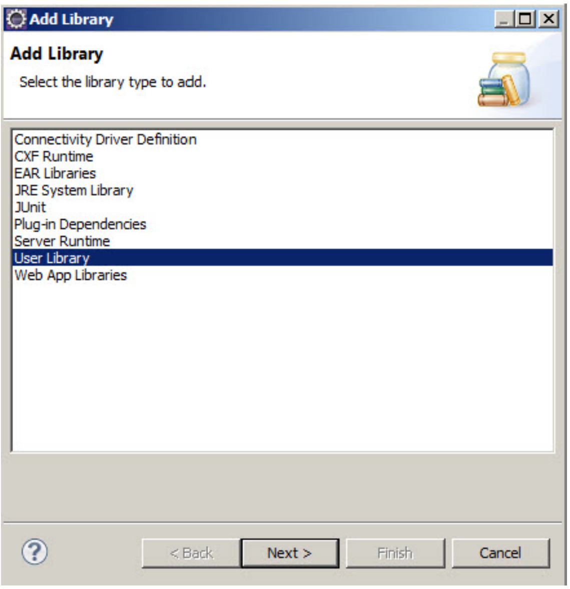


Connect to your task and ALM tools or create a local task

An outline is not available.

3. Select **User Library** item from the library type list, then click the **Next** button. The User Library page is displayed. See Figure 12.

Figure 12 The Add Library Dialog in Eclipse

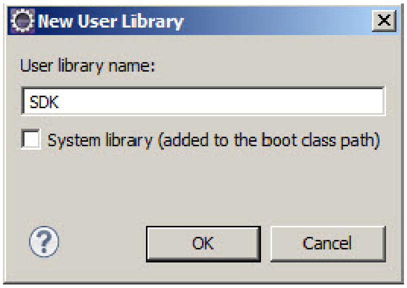


4. Click the **User Libraries...** button, the Preferences (Filtered) dialog is displayed.

5. Click the **New...** button in the right User Libraries panel, the New User Library dialog is displayed.

6. Enter the library name in the User Library Name text field, then click the **OK** button. See Figure 13.

Figure 13 Entering the User Library Name



User library name: SDK

□ System library (added to the boot class path)

7. Click the **Add JARs...** button, open the file: *TIBCO_HOME*`\adapter\sdk\`*version*`\lib\Maverick5.jar`. Click the **OK** button to exit the Preferences (Filtered) dialog.

8. Click the Finish button in the Add Library dialog, and the user library is created. See Figure 14 and Figure 15.

Figure 14 Finishing Adding the JAR File into the User Library

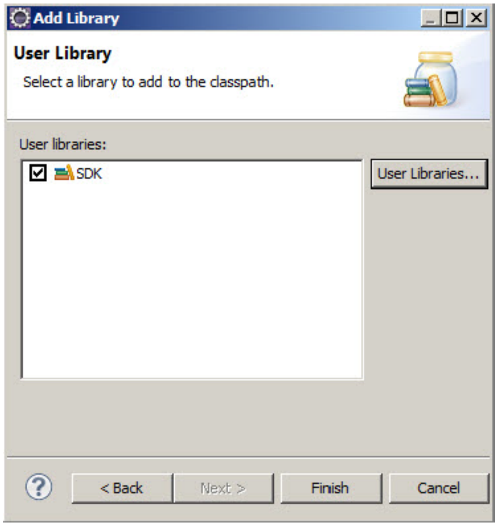
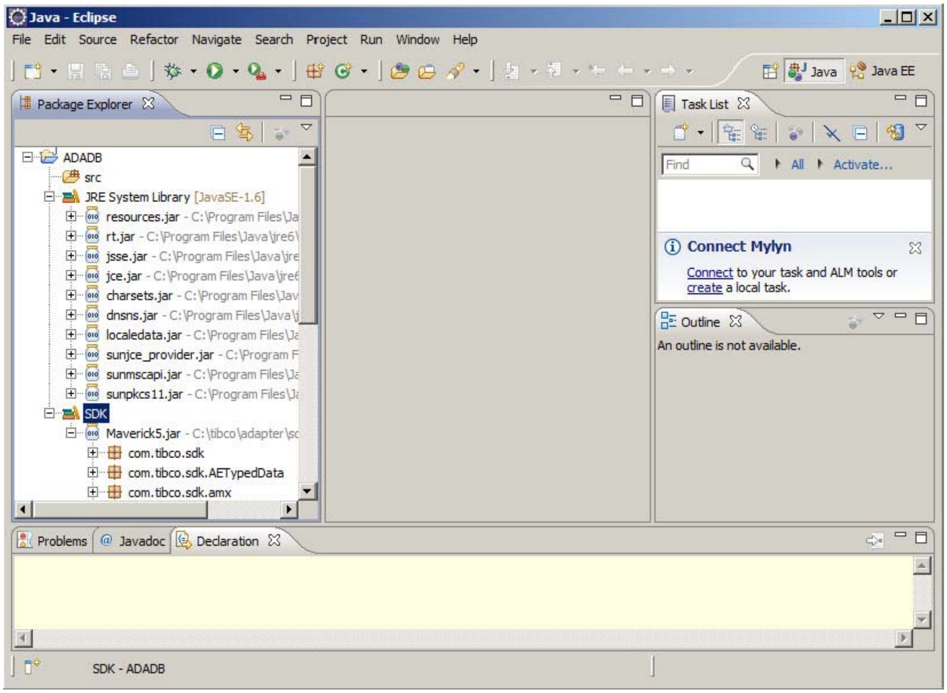


Figure 15 Finishing Adding the User Library

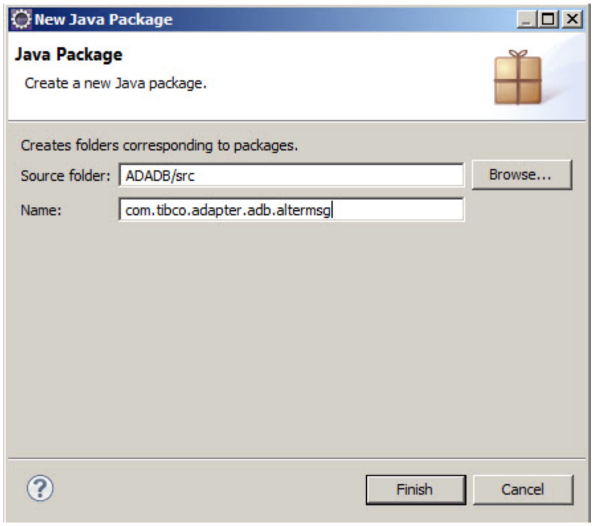


Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Java EE
Package Explorer
ADADB
src
JRE System Library [JavaSE-1.6]
resources.jar - C:\Program Files\Ja
rt.jar - C:\Program Files\Java\jre6\
jsse.jar - C:\Program Files\Java\jre
jce.jar - C:\Program Files\Java\jre6
charsets.jar - C:\Program Files\Jav
dnsns.jar - C:\Program Files\Java\j
localedata.jar - C:\Program Files\Ja
sunjce_provider.jar - C:\Program F
sunmscapi.jar - C:\Program Files\Ja
sunpkcs11.jar - C:\Program Files\Ja
SDK
Maverick5.jar - C:\tibco\adapter\sc
com.tibco.sdk
com.tibco.sdk.AETypedData
com.tibco.sdk.amx
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Problems
Javadoc
Declaration
SDK - ADADB

Adding a Java Package and Class

1. Right-click the **src** folder, then select **New > Package** from the menu that is displayed. The New Java Package dialog is displayed.
2. Enter the package name `com.tibco.adapter.adb.altermsg` in the Name text field, then click the **Finish** button. See Figure 16.

Figure 16 Adding a Java Package



New Java Package

Java Package

Create a new Java package.

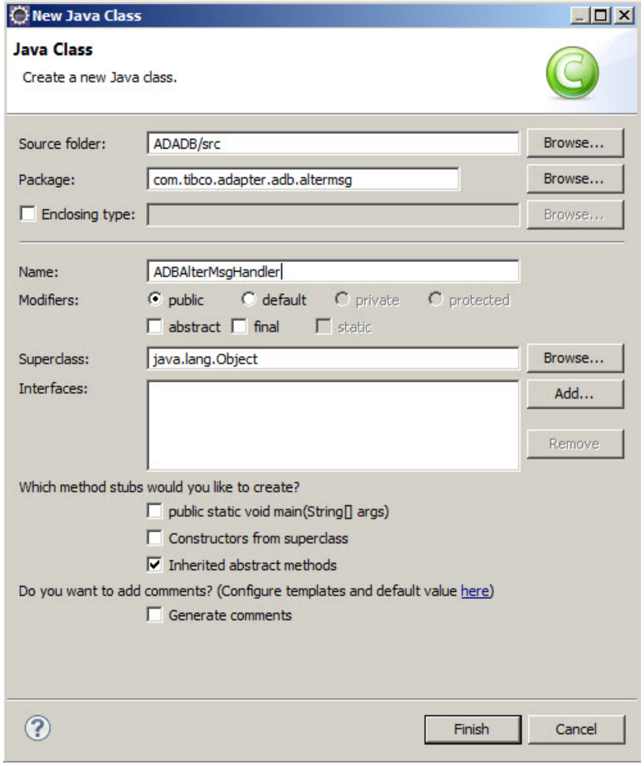
Creates folders corresponding to packages.

Source folder: ADADB/src

Name: com.tibco.adapter.adb.altermsg

3. Right-click the **`com.tibco.adapter.adb.altermsg`** item, then select **New > Class** from the menu that is displayed. The New Java Class dialog is displayed.
4. Enter the class name **`ADBAlterMsgHandler`** in the Name text field, then click the **Finish** button. See Figure 17.

Figure 17 Adding a Java Class



Java Class

Create a new Java class.

Source folder: ADADB/src

Package: com.tibco.adapter.adb.altermsg

□ Enclosing type:

Name: ADBAlterMsgHandler

Modifiers: ● public ○ default ○ private ○ protected

□ abstract □ final □ static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

□ public static void main(String[] args)

□ Constructors from superclass

☑ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

□ Generate comments

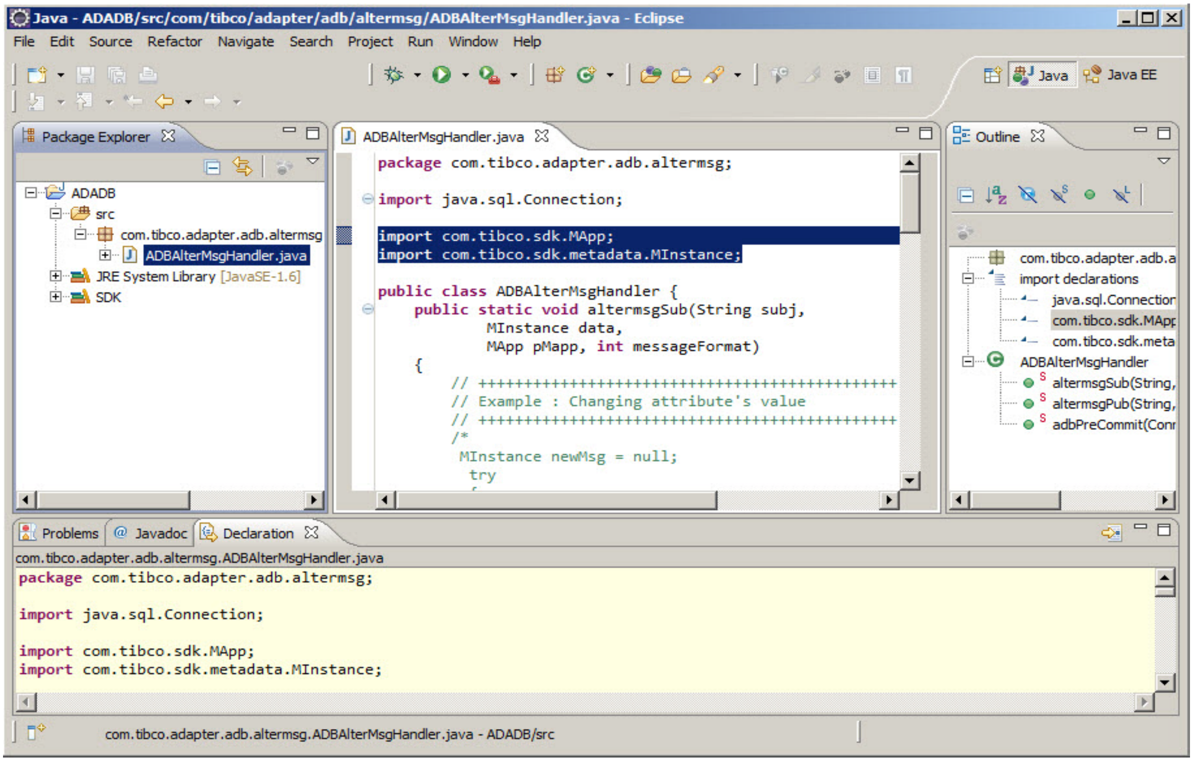
Using the alterMsgPub and alterMsgSub Functions

Two functions are available for creating callouts, `alterMsgPub()` to create a callout for a publisher and `alterMsgSub()` to create a callout for a subscriber. You can use the `alterMsgPub` and `alterMsgSub` functions to alter messages in the TIBCO ActiveEnterprise Message, TIBCO Rendezvous Message, and XML wire formats.

The `alterMsgPub()` and `alterMsgSub()` functions are the entry points into the callout library. These two function names and their signatures must not be changed. The function signatures are:

```
public static void altermsgPub(String subj,
             MInstance data,
             MApp pMapp, int messageFormat)

public static void altermsgSub(String subj,
             MInstance data,
             MApp pMapp, int messageFormat)
```

Figure 18 Using altermsgPub and altermsgSub Functions



```
package com.tibco.adapter.adb.altermsg;

import java.sql.Connection;

import com.tibco.sdk.MApp;
import com.tibco.sdk.metadata.MInstance;

public class ADBAlterMsgHandler {
    public static void altermsgSub(String subj,
            MInstance data,
            MApp pMapp, int messageFormat)
    {
        // +++++++++++++++++++++++++++++++++++++++++++++
        // Example : Changing attribute's value
        // +++++++++++++++++++++++++++++++++++++++++++++
        /*
         MInstance newMsg = null;
          try
```

com.tibco.adapter.adb.altermsg.ADBAlterMsgHandler.java

```
package com.tibco.adapter.adb.altermsg;

import java.sql.Connection;

import com.tibco.sdk.MApp;
import com.tibco.sdk.metadata.MInstance;
```

com.tibco.adapter.adb.altermsg.ADBAlterMsgHandler.java - ADADB/src

Both functions share the same set of parameters. However, several parameters are specific to a particular wire format.

Table 53 contains parameter descriptions and a column for each format. If the R column contains an X, use the parameter for TIBCO Rendezvous Message wire format. If the M column contains an X, use the parameter for TIBCO ActiveEnterprise Message or XML wire format. Otherwise, leave the value as NULL.

Table 53 Wire Format Parameters and Descriptions

Parameter	R	M	Description
subj	X	X	Subject on which the message is sent or received.
data	X		MInstance of the data. Alter directly if necessary.
pMapp		X	TIBCO Adapter SDK MApp structure for this MInstance.

Table 53 Wire Format Parameters and Descriptions (Cont'd)

Parameter	R	M	Description

messageFormat	X	X	The message format.
			Valid values are:
			`M_RV_MESSAGE_FORMAT`
			`M_AERV_MESSAGE_FORMAT`
			`M_XMLJMS_MESSAGE_FORMAT`
			`M_XMLRV_MESSAGE_FORMAT`
			Check TIBCO Adapter For SDK documentation for details of these value.

Using the adbPreCommit Function

The `adbPreCommit` function is used to perform a custom operation, such as invoking a stored procedure or sending a TIBCO Rendezvous message, just before the transaction is committed. This function is called by the subscriber after all inserts, updates and delete operations have been performed on both parent and child tables.

The function signature is:

```
public static int adbPreCommit(Connection connectHandle,
            MInstance data,
            MApp pMapp, int messageFormat)
```

If the return value is not zero, then the message will be rolled back.

Table 54 contains parameter descriptions and a column for each format.

Table 54 adbPreCommit Parameter Descriptions

Parameter	TIBCO Rendezvous Format	TIBCO ActiveEnterprise Format	Description
connectHandle	X	X	The database connection handle.
pMapp	X	X	TIBCO Adapter SDK MApp structure for this MInstance.

Table 54 adbPreCommit Parameter Descriptions (Cont'd)

Parameter	TIBCO Rendezvous Format	TIBCO ActiveEnterprise Format	Description

messageFormat	X	X	The message format.
			Valid values are:
			`M_RV_MESSAGE_FORMAT`
			`M_AERV_MESSAGE_FORMAT`
			`M_XMLJMS_MESSAGE_FORMAT`
			`M_XMLRV_MESSAGE_FORMAT`
			Check TIBCO Adapter For SDK documentation for details of these value.

If an adapter is configured to use the `adb.subBatchCommitTimeout` or `adb.subBatchCommitSize` options, the custom operation is performed once for each message received. However, the entire transaction is not committed until the size or time-out values are reached. Adapter configurations that use the bulk-insert-size option cannot use `adbPreCommit`.

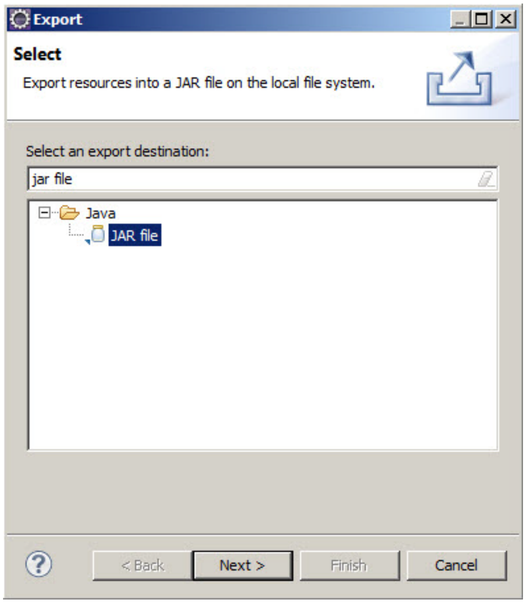
For more information on these options, see Table 57, Predefined Properties in TIBCO ActiveMatrix Adapter for Database, on page 264.

Exporting the JAR File

To export the project, follow these steps:

1. Right-click the project you want to export, then click the **Export...** item from the menu that is displayed. The Export dialog is displayed.
2. Type **`jar file`** in the Select an Export Destination text field, then click the **Next** button. The JAR Export dialog is displayed.

Figure 19 Typing Filter Text



Export resources into a JAR file on the local file system.

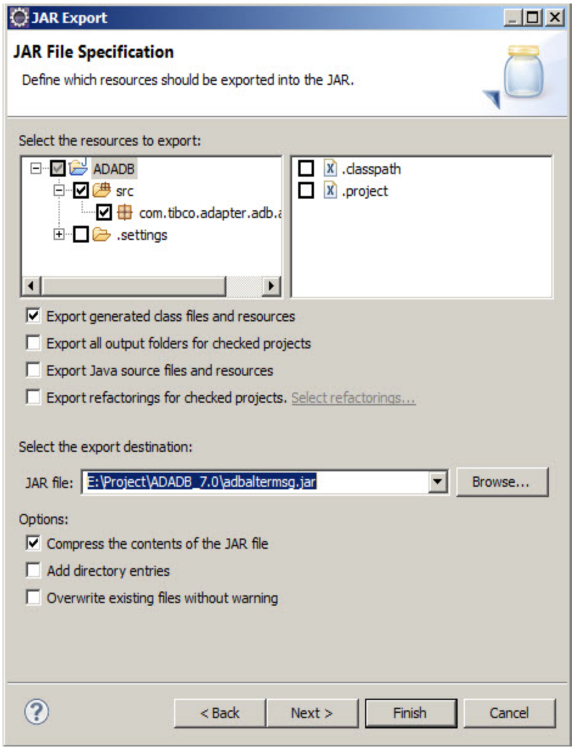
Select an export destination:

jar file

Java
JAR file

3. Select the resources to export, then type the export destination where you want to save the JAR file; or you can select a file repository by clicking the Browse... button. Click the **Finish** button.

Figure 20 Selecting the Export Destination



JAR Export

JAR File Specification

Define which resources should be exported into the JAR.

Select the resources to export:

- ☑ ADADB
  - ☑ src
    - ☑ com.tibco.adapter.adb.
  - □ .settings

- □ .classpath
- □ .project

☑ Export generated class files and resources

□ Export all output folders for checked projects

□ Export Java source files and resources

□ Export refactorings for checked projects. Select refactorings...

Select the export destination:

JAR file: E:\Project\ADADB_7.0\adbaltermsg.jar

Options:

☑ Compress the contents of the JAR file

□ Add directory entries

□ Overwrite existing files without warning

Replacing the Original JAR File

After exporting the JAR file, you need to use it to replace the original JAR file. To do so, follow these steps:

1. Open the adapter `lib` directory. For example:

   TIB_ADADB_HOME\`lib`

2. Backup the original `adbaltermsg.jar` file to another directory.
3. Copy and replace the original JAR file with the exported JAR file in the adapter `lib` directory.

If you no longer use the user callout library, you need to restore the original JAR file back to the adapter `lib` directory.

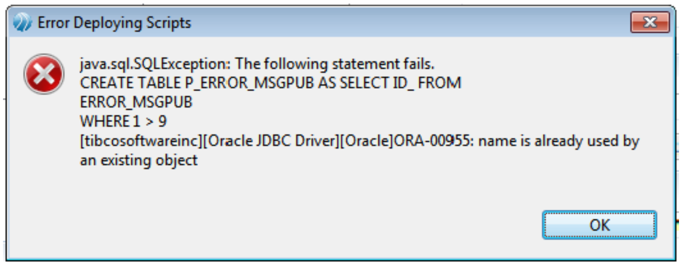
Using Database Deployment and Cleanup Scripts

Changing an existing adapter configuration typically generates legitimate changes in the database. During this change processes, the adapter creates a SQL script for changing the database objects and an associated cleanup script and stores them in the *TIB_ADADB_HOME*`\sql` directory. If the legitimate database changes result in error messages, you need to run these scripts. The messages and procedure for running the scripts are given in this section.

In the following cases, an error message as shown in Figure 21 will be displayed:

- If the scripts created for generating legitimate database changes are not successfully run.
- If the scripts created for generating legitimate database changes are not successfully run, but an error occurs while these changes are being saved to the database.

Figure 21 Database Error Message



```
java.sql.SQLException: The following statement fails.
CREATE TABLE P_ERROR_MSGPUB AS SELECT ID_ FROM
ERROR_MSGPUB
WHERE 1 > 9
[tibcosoftwareinc][Oracle JDBC Driver][Oracle]ORA-00955: name is already used by
an existing object
```

To Run the Deployment Scripts

The generated SQL script for changes to database objects is stored in the *instance_name*`.sql` file under the *TIB_ADADB_HOME*`\sql` directory. You can modify this script to deploy changes to different databases environment. For example, if the user schema is different between the production and testing environment, you can change the schema name of the database objects in the script and deploy the changes accordingly.

To Run the Cleanup Scripts

1. In TIBCO Business Studio, close the project containing the adapter configuration that you changed.
2. Fix the errors that caused the database changes to fail.

3. If necessary, clean up the old database configuration by running the scripts created by the adapter. The scripts are in the *TIB_ADADB_HOME*`\sql` directory and are named *instanceId*`.sql or` *instanceId*`.undo.sql`, where *instanceId* is the instance ID of the adapter configuration you changed.

4. Reopen the project in TIBCO Business Studio, and select the adapter configuration you were attempting to change when the errors occurred, then save the project.

OS Authentication

This release of the adapter provides SQL Server OS Authentication on Windows. Microsoft SQL Server uses integrated login security to establish connections using this data source, regardless of the current login security mode at the server. Any login ID or password supplied is ignored. The Microsoft SQL Server system administrator must have associated your Windows network ID with a Microsoft SQL Server login ID.

Runtime Schema

In the case where the schema name of the tables objects are different between development and production environment, you can use the following TRA properties to specify the schemas configuration.

For all database environments other than IBM UDB iSeries (AS400):

- `adb.originalSchema`—specify the design-time database object schema.
- `adb.runtime.schema`—specify the runtime table objects schema.
  - If `adb.originalSchema` is specified, the adapter will compare the `adb.originalSchema` with the prefix of the table objects. If they are the same, the adapter will replace the design-time schema with this runtime schema. Otherwise, no action will be taken.
  - If `adb.orignalSchema` is not specified, the adapter will append this runtime schema to all table objects that do not have any schema specified at design time.
  - If the name of the `adb.runtime.schema` parameter contains capital letter, then the name must be with the double quote. For example: `adb.runtime.schema="Test"`
- `adb.runtime.publisherSchema`—specify the runtime publisher table schema. This overrides the adb.runtime.schema setting. This can be used if the user has a different schema for the publishing table than the other table objects, such as source table.

These configurations do not apply to the referred object in publish by reference mode. User need to specify the runtime schema for the referred object during the design time, or in the `ADB_REF_OBJECT` column entry in the publishing table.

For IBM iSeries (AS400):

- `adb.as400.defaultLibrary`—specify the default iSeries library to be accessed.
- `adb.as400.library`—specify the name of the runtime iSeries library where the adapter will access.

Runtime Table Schema Configuration

You can specify the runtime schema global variables in the administrator console during the deployment of the adapter to enable the adapter to use these schemas in run time which are different from the design-time schema configuration. This makes it easy for you to move from one environment to another, most likely from the test environment to product environment.

For publisher by reference object, you need to make sure the right schema and reference object name are set correctly in the right publisher table, because the reference object name is fetched from the publisher table, not an property from the design-time project file or TRA file.

Specifying Query Timeout

The adapter provides query timeout on all databases. The operation for database returns with an error, instead of hanging if it cannot be completed when the database server is busy. For a Publication Service, the polling operation returns null and waits for the next polling. For a Subscription Service, it returns the connection dead error, and then reconnects.

Compressing JMS Messages (Publication Services only)

JMS message compression is an instance-level option. It is especially useful when messages are to be stored on the TIBCO Enterprise Message Service server, including persistent queue messages, or topics with durable subscribers. Enabling compression ensures that messages take less memory space in storage and are handled faster by the TIBCO Enterprise Message Service server. When JMS messages are compressed and stored, they are handled by the server in the compressed form.

The compression option only compresses the body of a message. Headers and properties are never compressed. It is best to enable compression when the message bodies are large and the messages are to be stored on a server.

When messages are not to be stored, compression is not as useful. Compression normally takes time, and therefore the time to send or publish and receive compressed messages is generally longer than the time to send the same messages uncompressed. There is little purpose to message compression for small messages that are not to be stored by the server.

You can enable or disable this feature for message senders with the following adapter property:

- `adb.jmscompress`: Either `ON` or `OFF`. The default value is `ON`, which indicates that messages will be compressed.

Configuring RVCMQ Backlog Size

The RVCMQ scheduler receives the inbound messages and assigns them to the worker. The scheduler stores tasks in a message queue. You can limit the maximum size of that queue with either one of the following two adapter properties or both of them:

- `adb.taskBackLogLimitInBytes`: specifies the maximum size of the scheduler task queue by number of bytes. This value must be an integer. The default value is unspecified.
- `adb.taskBackLogLimitInMessages`: specifies the maximum size of the scheduler task queue by number of messages. This value must be an integer. The default value is unspecified.

When the task messages in the queue exceed either of these limits, TIBCO Rendezvous deletes new inbound task messages.

If neither of the above properties are specified, there will be no limit to the size of the scheduler task queue.

RVCMQ backlog size configuration is for Subscription Services and Request-Response Services only, which have RVCMQ or RVDQ transport, not for Publication Services, which do not have this transport type.

Appendix A Frequently Asked Questions

This appendix lists answers to the frequently asked questions when configuring and deploying the adapter project.

Topics

General Questions

- Q1: How to find the version number of an adapter instance?
- Q2: Why is a database trigger error not logged in the exception table?
- Q3: How does the adapter react if the database connection is lost and the database is later restarted, and does it automatically try to reconnect?
- Q4: Does an incoming message contain all the columns that are defined for the destination table?
- Q5: Can an existing table be used as the publishing table?
- Q6: Does the TIBCO Rendezvous message that is published have to contain every field in the publishing table?
- Q7: If multiple updates occur between polling intervals, are updates published in multiple TIBCO Rendezvous messages or a single large message?
- Q8: Is it possible to delete older entries in the publishing table
- Q9: How does the exception table work?
- Q10: Can an adapter instance be used to replicate binary types, such as BLOB?
- Q11: Can an adapter instance write to tables that belong to a database account different from that used by the adapter?
- Q12: Can a publisher adapter and a subscriber adapter use different projects?
- Q13: Can TIBCO ActiveMatrix Adapter for Database guarantee exactly once delivery of messages over RVCM?
- Q14: What guarantees does the adbagent make with regards to the order of database operations? For instance, is it guaranteed that for a given table, modifications are made in the same order that they were made to the source database? What guarantees are made for operations across different tables?
- Q15: When using RVCM for delivery, at what point does the subscribing adapter acknowledge an incoming message?
- Q16: By maintaining the publishing tables, all changes to the source table can be captured. If, however, there is a failure between the point of publishing a message using RVCM and updating the publishing table, will the adapter republish the message that has already been sent?
- Q17: For the subscriber adapter, if failure occurs after doing a database update and before sending an acknowledgement back to the publisher adapter, can the restarted configuration be prevented from redoing the update operation?

- Q18: For a certified subscriber adapter, what happens if an insert exception occurs and no exception table is specified?
- Q19: How to preregister non-TIBCO ActiveMatrix Adapter for Database subscribers, like custom adapters, to ensure no messages are lost?
- Q20: The session names are automatically created. Can these be changed without confusing TIBCO ActiveMatrix Adapter for Database, so that we can use a standard naming convention throughout the entire integration process?
- Q21: What's the proper way to permanently remove an adbagent subscriber when using CM? One way is to completely REMOVE the publisher's ledger and to change all 'P' records back to 'N', then restart the publisher. Is there a better, more correct or automated method?
- Q22: Is it possible to run an adapter instance using a remote TIBCO Rendezvous daemon?
- Q23: Is it possible to run two configurations of the adapter on the same machine?
- Q24: Can an adapter instance collate information from several database tables to send as a single TIBCO Rendezvous message, or can it only publish data from a single table, in the format defined by that table?
- Q25: How to fix an adapter that hangs problem when my Sybase transaction log becomes full?

Q1: How to find the version number of an adapter instance?

A banner displays when an adapter instance starts. The banner lists component versions for the adapter and for TIBCO Adapter SDK software. You can use this information to diagnose compatibility issues, or to report any problem details to Customer Support.

Q2: Why is a database trigger error not logged in the exception table?

When using an adapter instance as a publisher, if an error occurs in the database trigger that is used to copy data from the source table to the publishing table, the database trigger error will not be logged in the exception table for the subscriber adapter.

Q3: How does the adapter react if the database connection is lost and the

database is later restarted, and does it automatically try to reconnect?

If TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) detects it has lost its database connection, it will connect to the database in three times by default. If the connection is not resumed after three times, the adapter exits. You can configure the adapter for automatic reconnection times and intervals (see Reconnection Configuration on page 67). Alternatively, TIBCO Hawk rules can be written to detect this and restart the adapter whenever this occurs.

Q4: Does an incoming message contain all the columns that are defined for the destination table?

The incoming message need not contain all the columns defined in the destination database table. You can configure the adapter to expect only a subset of the columns, defined in the repository. The adapter is driven from the subscribing class description and will iterate through the attributes in the class definition for the subscribing table and specifically look for those attributes in the incoming messages. It inserts NULLs for the attributes that it is expecting but does not find in the message. If there are more columns in the subscribing table than are listed in the subscribing class (set when adding a subscription), those extra columns will get whatever default values were specified during the table creation.

Q5: Can an existing table be used as the publishing table?

No. TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) requires additional columns in the publishing table. Even when every field in a table is published, a separate publishing table is required.

Q6: Does the TIBCO Rendezvous message that is published have to contain every field in the publishing table?

Yes. You can control which fields are copied to the publishing table by configuring the adapter and by changing the publication trigger to publish a subset of rows. You can also append additional fields to a message or drop a message based on some criteria using the user callout library. For more information on the user callout library, see Using the User Callout Java Library on page 213.

Q7: If multiple updates occur between polling intervals, are updates published in multiple TIBCO Rendezvous messages or a single large message?

If you are using `publish by value`, a TIBCO Rendezvous message is created for each individual update. If you are using `publish by reference`, that operation will get the last update.

Q8: Is it possible to delete older entries in the publishing table

Yes. When a row is published, the value of the ADB_L_DELIVERY_STATUS field in the publishing table changes to either `C` (complete) or `F` (failed). You can write a trigger in your publishing table that deletes the row when the delivery status changes to `C` or `F`.

You can also publish data directly from the source table by configuring the adapter instance to publish by reference. A publishing table is created, but it contains only required fields and key fields of the source table.

Q9: How does the exception table work?

Before starting an adapter instance, you must set the `adb.useExceptTable` option in the adapter's properties file to `on` and specify an exception table when configuring the adapter Subscription Service. If an error occurs when inserting data into the destination table, it will be inserted into the exception table. The transaction will be committed and a confirmation sent back for the message (RVCM delivery). If the insertion into the exception table also fails, an error message will display and the adapter instance will terminate.

Q10: Can an adapter instance be used to replicate binary types, such as BLOB?

There is only limited support for binary large object (BLOB) data types. Oracle LONG and LONG RAW types are supported in top-level tables when the adapter is configured to publish by reference. Oracle BLOB and CLOB data types are supported.

Q11: Can an adapter instance write to tables that belong to a database account different from that used by the adapter?

Yes. A source table or destination table can belong to a different database user than the default account created in `create_user.sql`. For more information on referencing external schemas, see Referencing an External Schema on page 83.

Q12: Can a publisher adapter and a subscriber adapter use different projects?

Yes, unless the publisher adapter is configured to use parent-child relationships.

Q13: Can TIBCO ActiveMatrix Adapter for Database guarantee exactly once

delivery of messages over RVCM?

Exactly once delivery of messages over RVCM is not currently guaranteed. The same quality of service that RVCM provides is supported, which is at least once. To get exactly once delivery requires combining the messaging operations and the database operations in a single atomic transaction, which is not supported in RVCM.

Q14: What guarantees does the adbagent make with regards to the order of database operations? For instance, is it guaranteed that *for a given table*, modifications are made in the same order that they were made to the source database? What guarantees are made for operations across different tables?

TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) guarantees that for database operations that are published on the same subject, the order of the operations is preserved. Usually, this applies to database operations made to one table. It does not usually apply to database operations across different tables.

Q15: When using RVCM for delivery, at what point does the subscribing adapter acknowledge an incoming message?

A subscriber adapter confirms the message only after the database operation is committed. If there is an error and no exception table is used, the database operation is rolled back and no confirmation is sent. If there is an error and an exception table is used, the insert to the exception table is committed and the message is then confirmed.

Q16: By maintaining the publishing tables, all changes to the source table can be captured. If, however, there is a failure between the point of publishing a message using RVCM and updating the publishing table, will the adapter republish the message that has already been sent?

Yes, the message will be republished and the subscriber would have to deal with the duplicate message.

Q17: For the subscriber adapter, if failure occurs after doing a database update and before sending an acknowledgement back to the publisher adapter, can the restarted configuration be prevented from redoing the update operation?

No. This will cause a duplicate insert.

Q18: For a certified subscriber adapter, what happens if an insert exception

occurs and no exception table is specified?

If the `tibco.clientVar.DirTrace` option is specified in the adapter's properties file when the adapter is started, exception handling information is written to the log file and the configuration continues to run. Since the insert could not be performed, the `ADB_L_DELIVERY_STATUS` publishing table column has a value of `P` for the message.

Q19: How to preregister non-TIBCO ActiveMatrix Adapter for Database subscribers, like custom adapters, to ensure no messages are lost?

Specify the `CM name` of the listener's RVCM session.

Q20: The session names are automatically created. Can these be changed without confusing TIBCO ActiveMatrix Adapter for Database, so that we can use a standard naming convention throughout the entire integration process?

No, the session names are fixed and used by TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio) internally. They cannot be altered.

Q21: What's the proper way to permanently remove an adbagent subscriber when using CM? One way is to completely REMOVE the publisher's ledger and to change all 'P' records back to 'N', then restart the publisher. Is there a better, more correct or automated method?

There is a TIBCO Hawk method, `unRegisterListener()`, which unregisters a CM subscription. This is the proper way to remove the adbagent subscriber as a CM listener.

Q22: Is it possible to run an adapter instance using a remote TIBCO Rendezvous daemon?

Yes. Change the default settings for network, service, and daemon parameters for the adapter using TIBCO Business Studio.

Q23: Is it possible to run two configurations of the adapter on the same machine?

It's possible to run multiple configurations of the adapter on the same machine if each adapter instance has a unique name. If both configurations use TIBCO Rendezvous certified messaging, each must use a different RVCM session.

Q24: Can an adapter instance collate information from several database tables to send as a single TIBCO Rendezvous message, or can it only publish data from a single table, in the format defined by that table?

There are two ways to publish related tables:

- Set the `adb.publishChildData` option to `on` when configuring the adapter's properties file. When there are insertions into parent table, the adapter will publish parent rows and the corresponding child rows using TIBCO ActiveEnterprise or XML format.

  The adapter currently does not support publishing child data in the TIBCO Rendezvous Message format. Update and delete on parent-child relationship when publishing is also not supported. See Publication Options Tab on page 103.

- Combine several tables into one table using a trigger, then publish from the combined table.

When publishing or subscribing, an adapter instance allows you to change the message using the callout library. See Using the User Callout Java Library on page 213 for details.

Q25: How to fix an adapter that hangs problem when my Sybase transaction log becomes full?

When a Sybase transaction log becomes full, if the database setting `abort trans on log full` is set to `false`, your application will hang instead of printing a transaction log full error. Execute the following command:

```
sp_dboption dbname, "abort tran on log full", true
```

Request-Response Service Questions

The following questions you may encounter when you are working on Request-Response Service:

- Q1: When using Request-Response Services, can an INSERT statement with values only (without field names) be sent to improve application's performance?
- Q2: Can an application send UPDATE statements to a subscriber adapter with only those fields which are being updated? That is, if a table has ten records and only two need to be updated, can an UPDATE statement be constructed and sent only for those fields?
- Q3: Does an adapter instance send responses back as one large message with all rows in it or is the message sent in chunks?
- Q4: What is returned to the application if a failure occurs when doing an insert or update?

Q1: When using Request-Response Services, can an INSERT statement with values only (without field names) be sent to improve application's performance?

Yes this is allowed. Your application can also send INSERT statements without the binds.

Q2: Can an application send UPDATE statements to a subscriber adapter with only those fields which are being updated? That is, if a table has ten records and only two need to be updated, can an UPDATE statement be constructed and sent only for those fields?

Yes, this is supported.

Q3: Does an adapter instance send responses back as one large message with all rows in it or is the message sent in chunks?

The adapter sends results back to an application as one large message.

Q4: What is returned to the application if a failure occurs when doing an insert or update?

If an error occurs while the adapter is processing a request, an error code and description is returned to the application. In the case of success, a result set and row count is returned to the application.

Appendix B Supported Data Types

This appendix lists the database data types supported by TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio).

If you select a database that contains BIT data types, the BIT data cannot be sent out using TIBCO Business Studio. In Java, the smallest data type, other than boolean, are bytes. Generally, a byte needs 8 bits of space to be stored. So the bit column in the database cannot be less than 8 bits of space.

Topics

Oracle

The following Oracle data types are supported:

Oracle Data Type	JDBC Data Type
BLOB	BLOB
CHAR	CHAR
CLOB	CLOB
DATE	TIMESTAMP
FLOAT(n)	DOUBLE
LONG	LONGVARCHAR
LONG RAW	LONGVARBINARY
NCHAR	CHAR or NCHAR[1]
NCLOB	CLOB or NCLOB[1]
NUMBER	DECIMAL
NUMBER(p, s)	DECIMAL
NVARCHAR2	VARCHAR or NVARCHAR[1]
RAW	VARBINARY
TIMESTAMP[2]	TIMESTAMP
TIMESTAMP WITH LOCAL TIMEZONE[2]	TIMESTAMP
TIMESTAMP WITH TIMEZONE[2] [3]	VARCHAR or TIMESTAMP
VARCHAR2[2]	VARCHAR
XMLType[2]	CLOB or SQLXM[1]

1. When `JDBCBehavior=0`, the data type depends on the JVM used by the application. For JVMs earlier than Java SE 6, the first value applies. For Java SE 6 and higher, the second value applies.

2. Supported only for Oracle 9i and higher.

3. When `FetchTSWTSasTimestamp=false` (default), this data type is mapped to the JDBC VARCHAR data type; when `FetchTSWTSasTimestamp=true`, it is mapped to the JDBC TIMESTAMP data type.

Microsoft SQL Server

The following Microsoft SQL Server data types are supported:

Microsoft SQL Server Data Type	JDBC Data Type
BIGINT[1]	BIGINT
BINARY	BINARY
BIT	BIT
CHAR	CHAR
DATETIME	TIMESTAMP
DECIMAL	DECIMAL
FLOAT	FLOAT
IMAGE	LONGVARBINARY
INT	INTEGER
MONEY	DECIMAL
NCHAR	CHAR or NCHAR[2]
NTEXT	LONGVARCHAR or LONGNVARCHAR[2]
NUMERIC	NUMERIC
NVARCHAR	VARCHAR or NVARCHAR[2]
REAL	REAL
SMALLDATETIME	TIMESTAMP
SMALLINT	SMALLINT
SMALLMONEY	DECIMAL
TEXT	LONGVARCHAR
TIMESTAMP	BINARY
TINYINT	TINYINT
UNIQUEIDENTIFIER	CHAR

Microsoft SQL Server Data Type	JDBC Data Type
VARBINARY	VARBINARY
VARCHAR	VARCHAR

1. Supported only for Microsoft SQL Server 2000 and higher.

2. When `JDBCBehavior=0`, the data type depends on the JVM used by the application. For JVMs earlier than Java SE 6, the first value applies. For Java SE 6 and higher, the second value applies.

- When you insert a table with TEXT, NTEXT, and IMAGE data type into a new project, the adapter fails to create the trigger, and the project cannot be saved.
- When you insert a table with MONEY, SMALLMONEY, DECIMAL, or NUMERICA data type into a project, if you request the numbers with high precision, you need to choose `string` instead of `r8` in AE Type in the Table tab.
- When creating a table that contains TIMESTAMP data type, the publishing tables cannot be created.

Sybase and Sybase Adaptive Server Anywhere

The following Sybase data types are supported:

Sybase Data Type	JDBC Data Type
BIGINT[1]	BIGINT
BINARY	BINARY
BIT	BIT
CHAR	CAHR
DATETIME	TIMESTAMP
DECIMAL	DECIMAL
FLOAT	FLOAT
IMAGE	LONGVARBINARY
INT	INTEGER
MONEY	DECIMAL
NUMERIC	NUMERIC
REAL	REAL
SMALLDATETIME	TIMESTAMP
SMALLINT	SMALLINT
SMALLMONEY	DECIMAL
TEXT	LONGVARCHAR
TINYINT	TINYINT
VARBINARY	VARBINARY
VARCHAR	VARCHAR

[1] Supported only for Sybase 15.0 and higher.

DB2

The following DB2 data types are supported:

DB2 Data Type	JDBC Data Type
BIGINT[1]	BIGINT
BLOB[2]	BLOB
CHAR	CHAR
CLOB	CLOB
DATE	DATE or TIMESTAMP[3]
DECIMAL	DECIMAL
DOUBLE	DOUBLE
FLOAT	FLOAT
GRAPHIC	CHAR or NCHAR[4]
INTEGER	INTEGER
LONG VARCHAR	LONGVARCHAR
LONG VARGRAPHIC	LONGVARCHAR or LONGNVARCHAR[4]
NUMERIC	NUMERIC
REAL	REAL
SMALLINT	SMALLINT
TIME	TIME
TIMESTAMP	TIMESTAMP
VARCHAR	VARCHAR

[1] Supported for DB2 V9.1, V10 for z/OS.

[2] Supported only for DB2 V8.1 and higher for Linux/UNIX/Windows, DB2 for z/OS, and DB2 V5R2 for iSeries.

[3] For DB2 V9.7 for Linux/UNIX/Windows with the Oracle compatibility feature enabled, the Date type maps to the JDBC TIMESTAMP type.

4. When `JDBCBehavior=0`, the data type depends on the JVM the application uses. For JVMs earlier than Java SE 6, the first value applies. For Java SE 6 and higher, the second value applies.

When users select the DB2 OS390 database from the Vendor drop-down list in the Configuration tab, be aware the following:

- When creating a table that includes these data types, BLOB, CLOB, and VARCHARC, you can create the publishing table by selecting Publish by Reference storage mode.
- When creating a table with the CLOB or BLOB as the data type, and the data maximum value exceeds 32K limit, then the Subscription Service will not accept the data.
- When creating procedure, datatype LONGVARCHAR and LONGVARGRAPHIC are not supported in DB2 for zOS.
- When creating an opaque exception table, you need to create a new LOB table space first, and then create the corresponding auxiliary table and the index.
- When you insert a table with DECIMAL or NUMERIC data type into a project, if you request the numbers with high precision, you need to choose `string` instead of `r8` in AE Type in the Table tab.

MySQL

The following MySQL data types are supported:

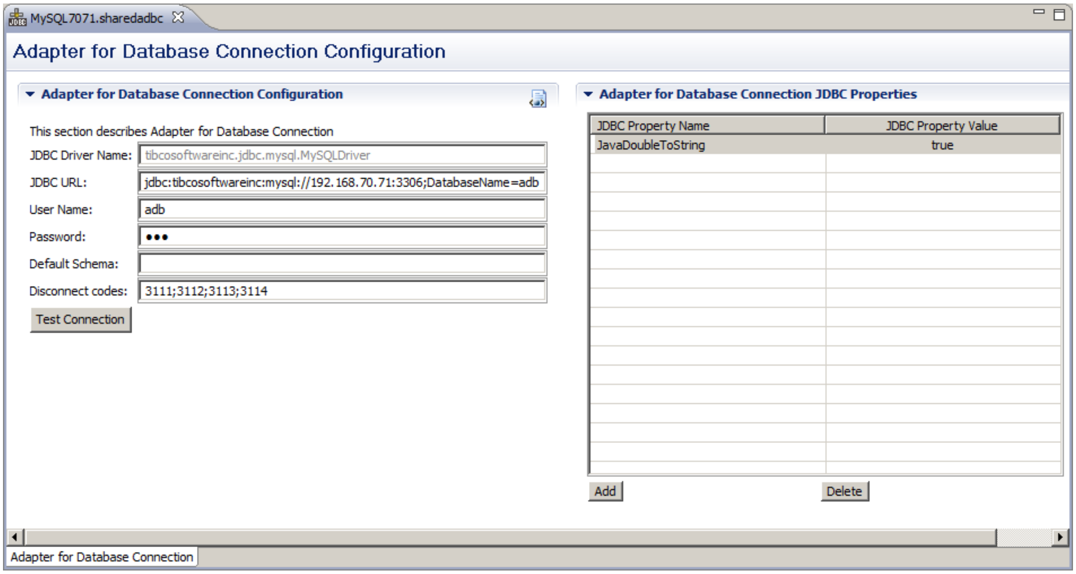
MySQL Data Type	JDBC Data Type
BIGINT	BIGINT
BIGINT UNSIGNED	BIGINT
BINARY	BINARY
BIT	BINARY
BLOB	LONGVARBINARY
CHAR(n)	CHAR
DATE	DATE
DATETIME	TIMESTAMP
DECIMAL	DECIMAL
DOUBLE	DOUBLE
FLOAT	REAL
INTEGER	INTEGER
LONGBLOB	LONGVARBINARY
LONGTEXT	LONGVARCHAR
MEDIUMBLOB	LONGVARBINARY
MEDIUMINT	INTEGER
MEDIUMINT UNSIGNED	INTEGER
MEDIUMTEXT	LONGVARCHAR
SMALLINT	SMALLINT
SMALLINT UNSIGNED	SMALLINT
TEXT	LONGVARCHAR
TIME	TIME

MySQL Data Type	JDBC Data Type
TIMESTAMP	TIMESTAMP
TINYBLOB	LONGVARBINARY
TINYINT	TINYINT
TINYINT UNSIGNED	TINYINT
TINYTEXT	LONGVARCHAR
VARBINARY	VARBINARY
VARCHAR(n)	VARCHAR
YEAR	SMALLINT

When running Request-Response Service with MySQL database, and set DOUBLE as the data type, you need to change the DOUBLE data type to the STRING data type to avoid a boundary value error occurs.

You can set the connection option in the Adapter for Database Connection Configuration. See Figure 22.

Figure 22 Setting Connection Option in Adapter for Database Connection Configuration



Adapter for Database Connection Configuration

Adapter for Database Connection Configuration

This section describes Adapter for Database Connection

JDBC Driver Name: tibcosoftwareinc.jdbc.mysql.MySQLDriver

JDBC URL: jdbc:tibcosoftwareinc:mysql://192.168.70.71:3306;DatabaseName=adb

User Name: adb

Password: •••

Default Schema:

Disconnect codes: 3111;3112;3113;3114

Test Connection

Adapter for Database Connection JDBC Properties

JDBC Property Name	JDBC Property Value
JavaDoubleToString	true

Teradata

The following Teradata data types are supported:

Teradata Data Type	JDBC Data Type
BYTE	BINARY
BYTEINT	TINYINT
CHAR	CHAR
DATE	DATE
DECIMAL	DECIMAL
FLOAT	FLOAT
INTEGER	INTEGER
NUMERIC	NUMERIC
REAL	REAL
SMALLINT	SMALLINT
TIME	TIME
TIME WITH TIME ZONE	TIME
TIMESTAMP	TIMESTAMP
TIMESTAMP WITH TIME ZONE	TIMESTAMP
VARBYTE	VARBINARY
VARCHAR	VARCHAR

PostgreSQL and EnterpriseDB

The following PostgreSQL and EnterpriseDB data types are supported:

PostgreSQL Data Type	JDBC Data Type
BIGINT	BIGINT
BOOL	BIT
BYTEA	BINARY
CHAR	CHAR
DATE	DATE
DECIMAL	DECIMAL
DOUBLE	DOUBLE
INT2	TINYINT, SMALLINT
INT4	INTEGER
INT8	BIGINT
NUMERIC	DECIMAL, NUMERIC
REAL	REAL
TIME	TIME
TIMESTAMP	TIMESTAMP
VARCHAR	VARCHAR, LONGVARCHAR

Appendix C Predefined Module Properties

Table 55 explains the predefined module properties of TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio). Some module properties are automatically used within the system when an adapter configuration is defined.

Table 55 Predefined Module Properties (Sheet 1 of 3)

Variable	Description
`ADBScriptFileDir`	Directory to store the `.sql` scripts for cleaning up the database. Do not include the `.sql` file name in the property value.
Deployment	Defaults to the TIBCO Designer project name. This value can be any string value. This global variable is used by the system to partially define the subject name defined for a service.
DirLedger	Specifies the path name of the TIBCO Rendezvous certified messaging ledger file. The default is the root installation directory.
DirTrace	Specifies the path name for log file used by the adapter. The default is the root installation directory.
Domain	The default value for file-based local projects is `Domain`. The value for server-based projects is the domain to which the project was saved.
JmsProviderUrl	Tells applications where the JMS daemon is located. Setting this value mostly makes sense in early stages of a project, when only one JMS daemon is used.
JmsSslProviderUrl	Tells applications where the JMS SSL daemon is located.
RemoteRvDaemon	TIBCO Rendezvous routing daemon (`rvrd`) to be used. See *TIBCO Administrator Server Configuration Guide* for details about setting up a domain using `rvrd`.
RvDaemon	TIBCO Rendezvous daemon. Sessions use this daemon to establish communication. The default value is `7500`.

Table 55 Predefined Module Properties (Sheet 2 of 3)

Variable	Description
RvNetwork	TIBCO Rendezvous network. This variable need only be set on computers with more than one network interface. If specified, the TIBCO Rendezvous daemon uses that network for all outbound messages. In most cases, you can leave the default.
RvService	TIBCO Rendezvous service. The TIBCO Rendezvous daemon divides the network into logical partitions. Each transport communicates on a single service. A transport can communicate only on the same service with other transports. Unless you are using a non-default TIBCO Rendezvous configuration, you should leave the default (`7500`).
RvaHost	Computer on which the TIBCO Rendezvous agent runs. This variable is only relevant if you are using the TIBCO Rendezvous Agent (`rva`) instead of the TIBCO Rendezvous daemon, and if you have configured a non-default setup. See *TIBCO Rendezvous Administration* for details about specifying the `rva` parameters.
RvaPort	TCP port where the TIBCO Rendezvous agent (`rva`) listens for client connection requests. See *TIBCO Rendezvous Administration* for details about specifying the `rva` parameters. Default is `7501`.
TIBHawkDaemon	TIBCO Rendezvous daemon used in the TIBCO Hawk session. Specifies which Hawk daemon handles communication for the session. A local daemon is specified by the communications type (always `tcp`) and a socket number. The default configuration uses the local daemon with the TCP socket number `7474`. Specify a remote daemon by inserting its host name or IP address between the `tcp` entry and the port number of the daemon parameter, such as `tcp:remote_computer:7800.` See *TIBCO Hawk Installation and Configuration* manual for details about this parameter.
TIBHawkNetwork	TIBCO Rendezvous network used by the TIBCO Hawk session. Specifies which network to use for outbound session communications when a computer is connected to more than one network, and also specifies the multicast groups to use for communication. See *TIBCO Hawk Installation and Configuration* manual for details about this parameter.

Table 55 Predefined Module Properties (Sheet 3 of 3)

Variable	Description
TIBHawkService	TIBCO Rendezvous service used by the TIBCO Hawk session. The `Service` parameter specifies which User Datagram Protocol (UDP) service group the TIBCO Rendezvous daemon should use for session communications. The default service port is `7474`. See *TIBCO Hawk Installation and Configuration* manual for details about this parameter.

Appendix D Adapter Properties File

The runtime adapter service parses a properties file at startup.

This appendix introduces the TRA properties file and the adapter properties defined in TIBCO ActiveMatrix Adapter for Database.

Topics

Overview of Adapter Properties File

The runtime adapter parses one or more properties file at startup.

The default runtime adapter properties file for TIBCO ActiveMatrix Adapter for Database is named `adbagent.tra`. The default properties file is located in the *TIB_ADADB_HOME*`\bin` subdirectory.

The adapter also provides a template TRA file for each properties file, when the actual properties file is corrupted or deleted by mistake.

Each line in a properties file is a single property. Each property consists of a key and a value. The key starts with the first non-whitespace character and ends at the first occurence of these characters:

(space) : =

The value starts at the first character after any of the three characters listed above.

For example:

```
tibco.configurl=/tibco/private/adapter/test/config/config1
tibco.repourl=tibcr://TEST_PROJECT
tibco.username=admin
tibco.password=samplePassword
tibco.clientVar.service=7600
tibco.clientVar.daemon=tcp:7600
```

Properties defined in the properties file of a deployed service override the same properties defined in the project. The properties file for a deployed service is located in the following directory:

TIBCO_TRA_DOMAIN_HOME/*domainName*/`application`/*applicationDeploymentName*

Properties File Format

The following restrictions apply to properties:

- Do not use the exclamation point (`!`) as a comment line indicator. Instead, use the number sign (#).
- The line continuation character is ignored, and you cannot define a property with multi-line values.
- A key cannot contain any of the termination characters. Although you can use termination characters by escaping the value with a preceding backslash (\) in Java, TIBCO ActiveMatrix Adapter for Database does not support this syntax.
- All paths inside a properties file, including Microsoft Windows directory names, must use forward slashes.

Password Handling

You can use the obfuscate utility installed with TIBCO Runtime Agent to encrypt confidential information, such as passwords in property files, with an encryption key. The obfuscate utility rewrites a Java property file by encrypting property values that start with a #!! or #! prefix. The utility is located in *TIB_TRA_HOME*`/bin`.

If you plan to run the adapter locally, define the runtime password value to be a global variable. Before starting the adapter, include the runtime password as a client variable in the properties file and obfuscate it using the obfuscate utility.

For example, follow these steps to encrypt the password:

1. Define the password as `myPassword` as a global variable with no value.
2. Include the following entry in the TRA file of the adapter at runtime:

   `tibco.clientVar.myPassword=#!!`*passwordValue*
3. Invoke the following command:

   TIB_TRA_HOME`/bin/obfuscate` *TRAFileName*

For more information about how to use the obfuscate utility, see "Obfuscate Utility" in *TIBCO Runtime Agent Installation*.

Adapter Properties

Properties are in two categories: Required Properties and TIBCO ActiveMatrix Adapter for Database Properties. This section explains these properties.

Required Properties

In order for a properties file to be used with a runtime adapter, the following properties must be correct for the configuration of the adapter.

Table 56 lists the properties required by a runtime adapter.

Most required properties are predefined by TIBCO Adapter SDK. See "Properties Files" in *TIBCO Adapter SDK Programmer's Guide* for more information.

Table 56 Required Runtime Adapter Properties File Parameters

Property	Description
`tibco.repourl` *repourl*	The absolute pathname to the local repository where the adapter configuration is defined. For a remote project, the *repourl* value should use the form `tibco.repourl tibcr@`*name* where *name* is the repository name. For example: `tibco.repourl tibcr@ADBRepoDefault` For Unix systems, the path separator includes a single forward slash (/. For example: `/local/tibco/repo/repo.dat`

`tibco.configurl` *relative_path*
or
`tibco.configurl` *absolute_path*
	The location of the adapter service inside the project file. If a relative path is specified, the adapter service is assumed to be under the default area in the project file (`/tibco/private/adapter/`). For example, the following value connects to an adapter service named `adbpub` in the `/tibco/private/adapter/` directory:
	`tibco.configurl adbpub`
	If an absolute path is specified, the adapter instance is looked up in the repository as defined by the argument. For example:
	`tibco:configurl /tibco/private/adapter/adbpub`

`tibco.instanceid` *instance name*	The name of the adapter instance. The length of the name cannot be larger than 80 characters.
`tibco.clientVar.adb.password`	The password to connect to the targeted database. This can be obfuscated using the instructions in Password Handling on page 262.

Table 56 Required Runtime Adapter Properties File Parameters (Cont'd)

Property	Description
`application.args`	The properties (`.tra`) file to pass to TIBCO ActiveMatrix Adapter for Database. For example: `application.args adbagent -system:propFile C:\tibco\adapter\adadb\7.0\bin\adbsub.tra`
`application.start.dir`	The pathname of the adapter to start. For example: `application.start.dir C:\tibco\adapter\adadb\7.0\bin\`

TIBCO ActiveMatrix Adapter for Database Properties

This section introduces the TRA properties predefined in TIBCO ActiveMatrix Adapter for Database. Properties that start with `ntservice` are available only on Microsoft Windows platforms. The following table is alphabetically sorted by the property name.

Table 57 Predefined Properties in TIBCO ActiveMatrix Adapter for Database (Sheet 1 of 11)

Property	Description
`adb.`*<publisher_service_name>*`.jmsProperties` *<property1>=<value1>*`,` *<property2>=<value2>*`,` *<property3>=<value3>*	Set JMS message properties when using JMS transport type for the Publication Service. For more information, see TIBCO Enterprise Message Service documentation. During the runtime, it has higher precedence than the `adb.jmscompress` variable.
`adb.`*<publisher_service_name>*`.lookback <on/off>`	When setting the value to on, the adapter can poll remaining records continuously regardless of the Polling Interval value you set.
`adb.`*<publiser_ service_name>*`.preRegisteredListeners` *<subjectName1:listenerName1*`,` *subjectName2:listenerName2>*	Preregisters RVCM names for the specified subjects. For example, `adb.ADBPublisher.preRegisteredListeners= adb.sub1:rvcm1,adb.sub2:rvcm2` preregisters RVCM name `rvcm1` to subject `adb.sub1`, and RVCM name `rvcm2` to subject `adb.sub2`. For more information, see Preregistering a Certified Subscriber on page 209.
adb.*<sessionName/serviceName>*`.ADBQueueSize` *<queue size>*	An internal message queue in the service level. The default value is `-1`, which means the adapter does not limit the queue size.

Table 57 Predefined Properties in TIBCO ActiveMatrix Adapter for Database (Sheet 2 of 11)

Property	Description
`adb.<`*tableName*`>.poll.hint <`*hint*`>`	This feature helps improve the performance of your queries. This feature is only supported by Oracle and SQLServer databases. See Using Hints (Publication Services only) on page 201 for more information.
`adb.addCustomHawkMethodsToClassMAgent <on/off>`	Setting the property to on to add custom methods to the adapter's standard microagent. Setting to `off` to disable adding. The default value is set to on.
`adb.as400.defaultLibrary <`*AS400 default libarary at design time*`>`	The default iSeries library to be accessed and only used for library name verification. Applies only to the Publication Service and Subscription Service.
`adb.as400.library <`*AS400 runtime library*`>`	It applies only to the Publication Service and Subscription Service. The rules for which library name is used are as follow: • If the library name specified in design time is equal to the name of `adb.as400.defaultlibrary`, the adapter will use `adb.as400.library` name specified in the TRA file. • If `adb.as400.library` name specified in the TRA file is equal to `adb.as400.defaultlibrary` name, the adapter will use the library name specified in design time. • If the library name is not defined in design time, the publishing table cannot be created.
`adb.batchPubStatusUpdates <on/off>`	The default property value is set to `off`. When `adb.PollingBatchSize` is used, this optimizes publishing performance by batching message status updates to the publishing table. Do not use this option when messages are published using a parameterized subject name. If an adapter instance stops before a batch update is performed, the status column is not updated. As a result, duplicate messages are published when the instance is restarted.

Table 57 Predefined Properties in TIBCO ActiveMatrix Adapter for Database (Sheet 3 of 11)

Property	Description
`adb.customScaleForNumberType` *<table>*`.`*<column>*`=`*<scale>*`,` *<table>*`.`*<column>*`=`*<scale>*	Applies only to Subscription Service. This property sets the default scale of Oracle Number(empty) datatype. It is used to control the entered number with high precision.
`adb.datePattern <date pattern for java.text.SimpleDateFormat, such as` *yyyy-MM-dd>*	The standard date format pattern for `java.text.SimpleDateFormat`. The default format is `yyyy-MM-dd`.
`adb.debug` *<level>*	The debug printing level. If not specified, the default, `2`, is used. Possible values are: `0` - Log no debug information. `1` - Log SQL commands executed against the database. `2` - Log binding data for each SQL command. `3` - Log all debug information.
`adb.disableTerminationSubject <on/off>`	When this property is set to on, the adapter does not terminate on receiving a termination subject message.
`adb.groupSize`*<publisher group size>*	Specifies the number of rows to publish in a single message. Overrides the `Group Size` setting in the publication service configuration.
`adb.jmsBytesMsg <on/off>`	Determine whether you use JMS transport type to send byte messages. The default value is `off`.
`adb.jmscompress <on/off>`	One of JMS properties. Applies only to the Publication Service with JMS transport type. Either on or `off`. The default value is `off`, which indicates that JMS messages is not compressed. See Compressing JMS Messages (Publication Services only) on page 230 for more information.
`adb.log4jPropFile <The path of the properties file>`	The path of the properties file.
`adb.maxLongLen -1 <maximum size of handle data. default value is -1, not check data length>`	The buffer size. Used for long data types such as `BLOB`.
`adb.noDupDetection <on/off>`	Disables detection of duplicate configurations. Either on or `off`. The default value is `off`, which indicates that duplicate configurations is detected.

Table 57 Predefined Properties in TIBCO ActiveMatrix Adapter for Database (Sheet 4 of 11)

Property	Description
`adb.originalSchema` *<design time database object schema name>*	The design-time table objects schema. For more information, see Runtime Table Schema Configuration on page 228.
`adb.password`	Password used by the adapter to access databases. If it is not specified, the runtime will call the password in the design time. If it is set in the TRA file, the runtime will call the password from the TRA file.
`adb.payloadOnError on|off`	This property specifies whether only the error information or all the logging information is printed in the log file. The default value is `off`, and it applies only to Subscription Service. When configuring this property, you need to pay attention to the following: • When creating the BusinessWorks process, you must set a value in the Reply Subject field in the Input tab. • If it is on or the Generate Payload On Error checkbox in the General tab is checked when configuring an adapter instance, and the Log Info Messages checkbox in the Logging tab is unchecked, the error log information that belongs to Debug role is displayed in Error role.
`adb.perf` *<the number of messages, it will print the performance report after process the number of message>*	The number of messages. It prints the performance report after processing the number of message.
`adb.perfMon <on/off>`	Either on or `off`. The default is `off`.
`adb.PollingBatchSize` *<number of rows>*	Applies only to publisher instances. Limits the amount of messages to be picked up. The value indicates the number of parent rows to fetch for a poll interval. The default value is `0`, which indicates that all new rows should be fetched.
`adb.PollingInterval` *<milliseconds>*	Specific polling period. Applies only to publisher instances. If not specified, the default value is `10000` milliseconds.
`adb.PollingCommitForDB2 <on/off>`	Either `on` or `off`. The default is `off`.

Table 57 Predefined Properties in TIBCO ActiveMatrix Adapter for Database (Sheet 5 of 11)

Property	Description
`adb.pubBatchConfirmSize` *<number of confirmations>*	Applies only to publisher instances with publications that use certified message delivery. Optimizes performance by batching message status advisories to the publishing table. The value indicates the number of advisory messages to include in a single batch. Do not use this option when messages are published using a parameterized subject name. **Note:** If an adapter instance stops before a batch update is performed, the status column is not updated. As a result, messages that were successfully published still have a status of `P` (pending) in the publishing table when the adapter instance is restarted. In this case, the ledger file contains the correct status information. A smaller value for this property decreases this risk.
`adb.pubBatchConfirmTimeout` *<milliseconds>*	Applies only to publisher instances with publications that use certified message delivery. This property specifies the number of milliseconds to wait before updating the status column. After this interval, an update is performed even if the batch size value is not reached. The default value is `10000` (10 seconds). A value of `0` means that no timeout interval is used. Do not use this option when messages are published using a parameterized subject name. **Note:** If an adapter instance stops before a batch update is performed, the status column is not updated. As a result, messages that were successfully published still have a status of `P` (pending) in the publishing table when the adapter instance is restarted. In this case, the ledger file contains the correct status information. A smaller value for this property decreases this risk.
`adb.publishChildData <on/off>`	Applies only to parent-child relationships in Publication Service. Enables publishing child table rows. By default, the value is set to on.
`adb.requestResponseMaxRows` *<number of maxRows>*	This property specifies the maximum number of rows to fetch. This can be used limit the memory usage of the adapter. The unfetched rows will be ignored by the adapter.
`adb.requestResponseThreads` *<the number of rpc default session threads>*	The number of threads used by the Request-Response Service. The default value is `1`.

Table 57 Predefined Properties in TIBCO ActiveMatrix Adapter for Database (Sheet 6 of 11)

Property	Description
`adb.RetryTotal` *<total reconnection attempts>*	Total number of reconnection attempts.
`adb.runtime.publisherSchema` *<runtime publisher table schema name>*	The runtime publisher table schema name. This overrides the `adb.runtime.schema` setting. This can be used if the user has a different schema for the publishing table than the other table objects, such as the source table. For more information, see Runtime Table Schema Configuration on page 228.
`adb.runtime.schema` *<runtime database object schema name>*	The runtime database object schema name. • If `adb.originalSchema` is specified, the adapter will compare the `adb.originalSchema` with the prefix of the table objects. If they are the same, the adapter will replace the design-time schema with this runtime schema. Otherwise, no action will be taken. • If `adb.orignalSchema` is not specified, the adapter will append this runtime schema to all table objects that do not have any schema specified at design time. For more information, see Runtime Table Schema Configuration on page 228.
`adb.rvAdvisoryNoLog <on/off>`	This property specifies whether RV advisory messages are be logged in the adapter log files. The default value is `off`.
`adb.rvMaxQueueSize` *<RV event queue size>*	Applies only to subscriber instances. Maximum number of messages to allow in the TIBCO Rendezvous event queue. The default value is `0`, which means no limit is placed on event queue size. **Note:** If setting the value less than the message count, the message may lose.
`adb.setClientInfo <Client info setting on Oracel>`	When set to on, the adapter will call `SET_CLIENT_INFO` to set the database session client information. The default value is on.

Table 57 Predefined Properties in TIBCO ActiveMatrix Adapter for Database (Sheet 7 of 11)

Property	Description
`adb.setEmptyStringNullForRvMsg <on/off>`	Either on or `off`. Specifies whether the RVMSG fields of an empty string are treated as `NULL` or `""` (empty strings). The default setting is `off`. If the property value is set to on, empty strings are treated as `NULL`. If the property value is set to `off`, empty strings are treated as empty strings if the database allows it.
`adb.setEmptyStringToDefault <on/off>`	Either on or off. The default setting is off. It only works when meeting the following three criteria: • in Oracle database • insert operations • set `JDBC Property CatalogOptions=1` When you set a default value to a field in creating table, if the property value is set to on, the empty strings are changed to the default value. **Note:** When setting the property `adb.setEmptyStringNullForRvMsg` to on, and selecting the Rendezvous Message wire format in the Subscription Service with Oracle database, the property `adb.setEmptyStringToDefault` becomes invalid.
`adb.SleepBetweenRetries` *<milliseconds of sleep between two reconnection attempts>*	Milliseconds of sleep between two reconnection attempts.
`adb.stmtCache` *<Number of statements to be cached>*	The number of cache statements for a generic RPC request/ reply service. The number of statements that the adapter caches will be executed directly for repeated requests. If the cache is full, the adapter will remove the oldest message from the cache and add the new statement. The default setting is `1`.
`adb.subBatchCommitSize`*<number of commit size>*	Applies only to subscriber instances. The number of messages to batch before invoking a commit operation. The default value is `0`. For more information, see Batch Processing in Subscription Services on page 199. **Note:** if messages greater than 32K are published, batching is automatically turned off.

Table 57 Predefined Properties in TIBCO ActiveMatrix Adapter for Database (Sheet 8 of 11)

Property	Description
`adb.subBatchCommitTimeout` *<milliseconds>*	Applies only to subscriber instances. The amount of time that can expire after a batch commit operation is invoked. If not specified, the default value is `10000` milliseconds. For more information, see Batch Processing in Subscription Services on page 199.
`adb.subBulkInsertSize` *<number of inserts>*	Applies only to subscriber instances. All incoming messages to insert are stored until this size is reached. Then a bulk insert operation is performed on the destination table. This value must be less than or equal to the value specified for `adb.subBatchCommitSize`, if used. The default value is `1`. If an update statement is published while messages are being batched, the bulk insert is performed regardless of whether the size value has been reached. After records have been inserted, the update operation is performed. For more information, see Batch Processing in Subscription Services on page 199. **Note:** Do not use this option if LONG, LONG RAW, image, or varbinary records are published.
`adb.taskBackLogLimitInBytes` *<The maximum size (in bytes) of the scheduler task queue>*	The scheduler stores tasks in a queue. This property limits the maximum size of that scheduler task queue by the number of bytes. The properties can control the memory usage on the adapter side. This value must be an integer. The default value is unspecified, which indicates that there is no limit to the size of the scheduler task queue.
`adb. taskBackLogLimitInMessages` *<The maximum size (in messages) of the scheduler task queue>*	The scheduler stores tasks in a queue. This property limits the maximum size of that scheduler task queue by the number of messages. The properties can control the memory usage on the adapter side. This value must be an integer. The default value is unspecified, which indicates that there is no limit to the size of the scheduler task queue.
`adb.terminateOnPubFail <on/off>`	Specifies that if publication fails during the message delivery, the agent will terminate after the status has been updated to ’F’. The default value is `off`.

Table 57   Predefined Properties in TIBCO ActiveMatrix Adapter for Database (Sheet 9 of 11)

Property	Description
`adb.timePattern <time pattern for java.text.SimpleDateFormat, such as` *HH:mm:ss>*	The standard time format pattern for `java.text.SimpleDateFormat`. The default format is `HH:mm:ss`.
`adb.timestampPattern <timestamp pattern for java.text.SimpleDateFormat, such as` *yyyy-MM-dd HH:mm:ss.S>*	The standard timestamp format pattern for `java.text.SimpleDateFormat`. The default format is `yyyy-MM-dd HH:mm:ss.S`.
`adb.useBetweenClause <on/off>`	This property disable the use of the `BETWEEN` clause in the select query of the publisher. The default value is on.
`adb.useExceptTable <on/off>`	Enables the use of the exception table. The exception table is defined when a subscription is created. The default value is set to on.
`adb.url`	Database url used by the adapter to access database. For example: `jdbc:tibcosoftwareinc:db2://IP:50000;databaseName=adb`
`adb.user`	Database account name used by the adapter to access databases. If it is not specified, the runtime will call the user name in the design time. If it is set in the TRA file, the runtime will call the user name from the TRA file.
`adb.verbose <`**`on/off`**`>`	Verbose mode. Print all available information to the console window or log file location. By default, the verbose mode is on.
`ntservice.account`	Username under which to run the Windows Service. You can use this property to initially set the account for the service, but once the service is installed, use the `Services` control to change the user account of services.

`ntservice.binary.path.absolute`	Absolute path to the executable that is run when the service is started. For example:
	`ntservice.binary.path.absolute`
	`   C:/tibco/adapter/adadb/7.0/bin/adbagent.exe`

`ntservice.dependencies`	The number of dependencies.

Table 57 Predefined Properties in TIBCO ActiveMatrix Adapter for Database (Sheet 10 of 11)

Property	Description
`ntservice.displayname`	Name to display in the Services control for this Windows Service. This property is useful if you want to have multiple Windows Services for the same executable. That is, you want to have two adapter running on the same machine. By specifying different service names and display names for the adapters, you can accomplish this.
`ntservice.interactive` **`true|false`**	Either `true` or `false`. Specifies whether the Windows Service is interactive. Set to `false` if you are not using a system account.
`ntservice.name`	Name for this Windows Service. This property is useful if you want to have multiple Windows Services for the same executable. That is, you want to have two adapters running on the same machine. By specifying different service names and display names for the adapters, you can accomplish this. For example: `ntservice.name` *adapter_instance_name*
`ntservice.password`	Password for the username in the `ntservice.account` property. You can use this property to initially set the password for the user account, but once the service is installed, use the `Services` control to change the password.
`ntservice.starttype` **`manual|automatic`**	Start type for this Windows Service. Either `manual` or `automatic`. For example: `ntservice.starttype automatic` You can use this property to initially set the start type for the service, but once the service is installed, use the Windows Services control to change the start type of services.

Table 57 Predefined Properties in TIBCO ActiveMatrix Adapter for Database (Sheet 11 of 11)

Property	Description
`tibco.clientVar`	Runtime values for global variables defined inside the repository. This value takes precedence over any global value set in the repository. Substitution takes place at runtime. You append the global variable to `tibco.clientVar`, then give its value. For example, a global variable named `DirLedger` is specified as follows: `tibco.clientVar.DirLedger C:/tibco/adapter/adadb/6.3/myledger` Do not include the `%` character of substitution variables. For example, to set `%%RvDaemon%%="tcp:7500"`, use `tibco.clientVar.RvDaemon "tcp:7500"`.
`tibco.username` `tibco.password`	Username and password used by the repository server to access the project. The password can be obfuscated using the instructions in Password Handling on page 262.
`-version`	Displays a banner with version information, then exits. This option is for troubleshooting purposes only.

Appendix E Trace Messages

This appendix explains the trace messages that are logged to a location specified at configuration time in TIBCO ActiveMatrix Adapter for Database (TIBCO Business Studio).

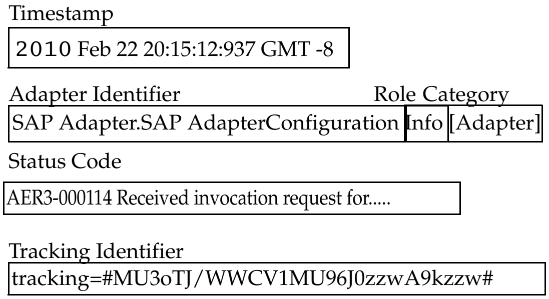
Topics

Trace Messages Overview

Trace messages provide information about adapter activities. The messages are logged to the console where the runtime adapter was 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 type.

Each trace message can include the following fields in the order shown:
`<Timestamp> <Adapter Identifier> <Role> <Category> <Status Code> <Tracking Identifier>`

The above fields are explained in Trace Message Fields on page 277. The following diagram shows an example trace message and calls out the fields.



```
2010 Feb 22 20:15:12:937 GMT -8
SAP Adapter.SAP AdapterConfiguration Info [Adapter]
AER3-000114 Received invocation request for.....
tracking=#MU3oTJ/WWCV1MU96J0zzwA9kzzw#
```

Example Trace Messages

The following trace messages are written during a session where TIBCO Adapter for Files received an object from TIBCO Adapter for R/3, then processed the object.

Example 1: Adapter Started

The following message indicates that TIBCO Adapter for Files 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 has started successfully.

```
2003 Feb 22 20:14:51:718 GMT -8
FileAdapter.FileAdapterConfiguration Info [Configuration]
AEFA-000058 TIBCO Adapter for Files successfully initialized
```

Example 2: Message Received

The next set of trace messages indicates the adapter has received an object that is sent on the TIBCO Rendezvous subject, `FROM.SAP`. The `#MU3oTJ/WWCV1MU96J0zzwA9kzzw#` tracking identifier that is included in the trace message uniquely identifies the message. The adapter (TIBCO Adapter for R/3) from which the message originated provided the identifier.

```
2003 Feb 22 20:15:12:937 GMT -8
FileAdapter.FileAdapterConfiguration Info [Adapter] AEFA-000067
Message containing class /tibco/public/class/ae/Customer received
on subject FROM.SAP tracking=#MU3oTJ/WWCV1MU96J0zzwA9kzzw#

2003 Feb 22 20:15:12:937 GMT -8
FileAdapter.FileAdapterConfiguration Info [Adapter] AEFA-000068
Message containing class /tibco/public/class/ae/Customer written
to working file customers.txt in Working Directory
F:\ca\integration\001\data_sets\files\wip
tracking=#MU3oTJ/WWCV1MU96J0zzwA9kzzw#
```

Example 3: Object Moved

The final trace message indicates the object has been moved to the output directory, which completes the adapter's interaction with the object. Because the trace message is the termination point, the tracking identifier is not displayed.

```
2003 Feb 22 20:15:42:812 GMT -8
FileAdapter.FileAdapterConfiguration Info [Adapter] AEFA-000070
File customers.txt is moved to the Output Directory
F:\ca\integration\001\data_sets\files\solutions\output
```

Trace Message Fields

Each trace message includes the following fields:

Table 58 Trace Message Fields (Sheet 1 of 3)

Field Name	Description
Timestamp	Timestamp of occurrence. For example, `2003 Feb 22 20:14:51:718 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. For example, the application identifier, `ADB.publisher1` identifies a TIBCO ActiveMatrix Adapter for Database service named publisher1.

Table 58 Trace Message Fields (Sheet 2 of 3)

Field Name	Description
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 is found. Processing continues, but special attention from an administrator is recommended. • **Error** Indicates an unrecoverable error occurs. Depending on the error severity, the adapter may continue with the next operation or may stop altogether. • **Debug** Indicates a developer-defined tracing message. In normal operating conditions and debug messages do not display. When configuring the adapter, you can define what roles need or need not to be logged. For example, in order to increase performance you can decide not to log Info roles.
Category	The following lists all categories : • **Adapter** The adapter is processing an event. • **Application** The adapter is interacting with the database. • **Configuration** The adapter is reading configuration information. • **Database** The adapter is interacting with a database. • **Metadata** The adapter is retrieving metadata from the database. • **Palette** The adapter is interacting with the palette. • **Publisher Service** The Publication service is reporting this trace message. • **Request-Response Client Service** The Request-Response Invocation service is reporting this trace message. • **Request-Response Server** The Request-Response Service is reporting this trace message. • **Shutdown** The adapter is shutting down. • **Startup** The adapter is starting. • **Subscription Service** The Subscription service is reporting this 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.

Table 58 Trace Message Fields (Sheet 3 of 3)

Field Name	Description
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 Trace Messages Reference on page 280 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.
Application Information	Application-specific information added to the tracking info to trace the message back to its source. Set initially by the originating adapter and carried forward. It is augmented by each intermediate component.

Trace Messages Reference

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

Resolutions are provided wherever possible for error and warning messages. If there is no resolution provided, or if you need additional help, contact TIBCO Support at http://support.tibco.com.

Status Code	Role	Category	Resolution
AEADB-100001	**Adapter starts to initialize.**		
	infoRole	Adapter	Normal operation; no action is necessary.
AEADB-100002	**Start %1 <%2>.**		
	infoRole	Adapter	Normal operation; no action is necessary.
AEADB-100003	**Start %1 <%2> with %3 session.**		
	infoRole	Adapter	Normal operation; no action is necessary.
AEADB-100004	**Adapter has started successfully.**		
	infoRole	Adapter	Normal operation; no action is necessary.
AEADB-100005	**Shutdown the adapter, current status is %1.**		
	infoRole	Adapter	Check database connect configuration or other configuration.
AEADB-100006	**Start to reconnect.**		
	infoRole	Adapter	Check whether the database connected or not.
AEADB-100007	**Reconnect successfully on attempt %2..**		
	infoRole	Adapter	Normal operation; no action is necessary.
AEADB-110001	**There is a same instance startup in the ip: <%1>.**		
	warnRole	Adapter	Check if the same instance name is started by another agent.

Status Code	Role	Category	Resolution (Cont’d)
AEADB-110002	**Received duplication detection request from ip: <%1>, host:<%2>.**		
	warnRole	Adapter	Normal operation; no action is necessary.
AEADB-110003	**No service type matched with the service <%1>.**		
	warnRole	Adapter	Check the transport type or the service type.
AEADB-110004	**Adapter thread <%1> the field <%2> size <%3> is larger than adb.maxLongLen <%4>.**		
	warnRole	Adapter	If you are not sure that you want to stop the adapter, you need to check the data length. To stop the adapter, set the `adb.maxLongLen` value smaller than the data length; otherwise, set it to `-1`.
AEADB-123001	**TIBCO Hawk agent is found and agent implementation has been registered.**		
	debugRole	Adapter	Normal operation; no action is necessary.
AEADB-123002	**Adapter connection test succeeded in startup.**		
	debugRole	Adapter	Normal operation; no action is necessary.
AEADB-123003	**Adapter configuration is parsed.**		
	debugRole	Adapter	Normal operation; no action is necessary.
AEADB-123004	**Adapter starts duplicate instance detection with the session <%1>.**		
	debugRole	Adapter	Normal operation; no action is necessary.
AEADB-123005	**Create thread <%1> for %2 <%3>.**		
	debugRole	Adapter	Normal operation; no action is necessary.
AEADB-123006	**Adapter termination event listener is created with session.**		
	debugRole	Adapter	Normal operation; no action is necessary.
AEADB-123007	**Monitor advisory %1 with session <%2>.**		
	debugRole	Adapter	Normal operation; no action is necessary.

Status Code	Role	Category	Resolution (Cont'd)
AEADB-123008	**Adapter reconnection thread is started.**		
	debugRole	Adapter	Check whether the database connected or not.
AEADB-123009	**Create dispatcher for session %1.**		
	debugRole	Adapter	Normal operation; no action is necessary.
AEADB-123010	**Publication Service <%1> preRegisteredListeners <%2> registered successfully.**		
	debugRole	Adapter	Normal operation; no action is necessary.
AEADB-123011	**Subscription Service Thread <%1> CustomScaleForNumberType set successfully for Table Columns: %2.**		
	debugRole	Adapter	Normal operation; no action is necessary.
AEADB-123012	**%1 <%2> Catalog:**		
	debugRole	Adapter	Normal operation; no action is necessary.
AEADB-130001	**Adapter throw exception %1 when shut down adapter.**		
	errorRole	Adapter	Check the error message and the configuration.
AEADB-130002	**Adapter failed to create termination event listener with session <%1>.**		
	errorRole	Adapter	Check the configuration.
AEADB-130003	**Adapter failed to reconnect.**		
	errorRole	Adapter	Check whether the database connected or not.
AEADB-130004	**Adapter duplication detection error: %1.**		
	errorRole	Adapter	Check if the message is null.
AEADB-130005	**Adapter connection test failed in startup.**		
	errorRole	Adapter	Check the configuration in the Design-time Connection tab.

Status Code	Role	Category	Resolution (Cont’d)
AEADB-200001	**Adapter receives one message from termination subject <%1>.**		
	infoRole	Adapter	Check if the message is MdataEvent or MexceptionEvent.
AEADB-200002	**Adapter will shutdown.**		
	infoRole	Adapter	Check the logs and the configuration.
AEADB-200003	**Adapter termination subject is disabled, so keep running.**		
	infoRole	Adapter	Normal operation; no action is necessary.
AEADB-300001	**Publication Service <%1> thread <%2> starts polling message.**		
	infoRole	Publication Service	Normal operation; no action is necessary.
AEADB-300002	**Publication Service <%1> thread <%2> publishes message on %3.**		
	infoRole	Publication Service	Normal operation; no action is necessary.
AEADB-300003	**Publication Service <%1> thread <%2> publishes group message on %3.**		
	infoRole	Publication Service	Normal operation; no action is necessary.
AEADB-300004	**Start alerter for Publication Services.**		
	infoRole	Publication Service	Normal operation; no action is necessary.
AEADB-300005	**Adapter handles alerter message.**		
	infoRole	Publication Service	Normal operation; no action is necessary.
AEADB-300006	**Publication Service <%1> thread <%2> add one row into the group message.**		
	infoRole	Publication Service	Normal operation; no action is necessary.

Status Code	Role	Category	Resolution (Cont'd)
AEADB-321001	**Publication Service <%1> execute rvcm Advisory Updater...**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-321002	**Publication Service <%1> RVCM Batch confirm %2 records.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-321003	**Publication Service <%1> Activate the Polling Timer.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-321004	**Publication Service <%1> Deactivate the Polling Timer.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-321005	**Publication Service <%1> thread <%2> execute publishing table mini sequence selector...**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-321006	**Publication Service <%1> thread <%2> execute (ADB_L_DELIVERY_STATUS or ADB_L_DELIVERY) N -> S marker...**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-321007	**Publication Service <%1> thread <%2> execute publishing table selector...**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-321008	**Publication Service <%1> thread <%2> execute reference object selector...**		
	debugRole	Publication Service	Normal operation; no action is necessary.

Status Code	Role	Category	Resolution (Cont'd)
AEADB-321009	**Publication Service <%1> thread <%2> execute pub table updater...**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-321010	**Publication Service <%1> thread <%2> polling commit for DB2.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-321011	**Publication Service <%1> thread <%2> batch update %3 records.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-321012	**Publication Service <%1> thread <%2> polls records without polling timer fired...**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-321013	**Publication Service <%1> thread <%2> execute fault status updater...**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-321014	**Publication Service <%1> thread <%2> execute 'S' to 'N' updater...**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-321015	**Execute listen_alert statement for alerter.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-322001	**Publication Service %1 thread %2 Fetch child data from %3.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-322002	**Publication Service %1 thread %2 execute child selector...**		
	debugRole	Publication Service	Normal operation; no action is necessary.

Status Code	Role	Category	Resolution (Cont’d)
AEADB-322003	**Publication Service <%1> thread <%2> selects record data from source table.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-322004	**Publication Service <%1> thread <%2> selects record data from reference objecct table.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-322005	**Publication Service <%1> thread <%2> Start to fetch data from child table.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-322006	**Publication Service <%1> thread <%2> Finish fetching data from child table.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-322007	**Publication Service <%1> thread <%2>: no record found...**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323001	**Publication Service %1 thread %2 Publishing Message Info:**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323002	**Publication Service <%1> begin to initialization.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323003	**Publication Service <%1> create data handler <%2>.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323004	**Publication Service <%1> data handler <%2> starts...**		
	debugRole	Publication Service	Normal operation; no action is necessary.

Status Code	Role	Category	Resolution (Cont'd)
AEADB-323005	**Publication Service <%1> create MTimer for rvcm batch confirm.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323006	**Publication Service <%1> create MTimer for polling.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323007	**Publication Service <%1> end to initialization.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323008	**Create alerter for Publication Services.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323009	**Initialize alerter for Publication Services.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323010	**Configure alerter of %1 for Publication Services.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323011	**Build listen_alert statement for alerter.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323012	**Publication Service <%1> Activate the RVCM batch confirm timer.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323013	**Publication Service <%1> Deactivate the RVCM batch confirm timer.**		
	debugRole	Publication Service	Normal operation; no action is necessary.

Status Code	Role	Category	Resolution (Cont'd)
AEADB-323014	**Publication Service <%1> Reset RVCM batch confirm timer.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323015	**Publication Service <%1> thread <%2> builds ADB_L_DELIVERY_STATUS or ADB_L_DELIVERY (DB2) N->S marker.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323016	**Publication Service <%1> thread <%2> builds publishing table selector whose ADB_L_DELIVERY_STATUS or ADB_L_DELIVERY (DB2) is S.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323017	**Publication Service <%1> thread <%2> builds publishing table mini sequence selector.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323018	**Publication Service <%1> thread <%2> binds (ADB_L_DELIVERY_STATUS or ADB_L_DELIVERY) N -> S marker...**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323019	**Publication Service <%1> thread <%2> builds publishing table selector.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323020	**Publication Service <%1> thread <%2> binds publishing table selector, polling batch size is %3.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323021	**Publication Service <%1> thread <%2> constructs the instance based on the publishing table.**		
	debugRole	Publication Service	Normal operation; no action is necessary.

Status Code	Role	Category	Resolution (Cont'd)
AEADB-323022	**Publication Service <%1> thread <%2> constructs the instance based on the source table.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323023	**Publication Service <%1> thread <%2> builds reference object selector.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323024	**Publication Service <%1> thread <%2> bind reference object selector.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323025	**Publication Service <%1> thread <%2> construct publisher endpoint.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323026	**Publication Service <%1> thread <%2>: the group size is %3, the current size is %4.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323027	**Publication Service <%1> thread <%2> builds pub table updater.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323028	**Publication Service <%1> thread <%2> binds pub table updater.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323029	**Publication Service <%1> thread <%2> acquire multithreading semaphore...**		
	debugRole	Publication Service	Normal operation; no action is necessary.

Status Code	Role	Category	Resolution (Cont'd)
AEADB-323030	**Publication Service <%1> thread <%2> release multithreading semaphore after marking records...**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323031	**Publication Service <%1> builds rvcm advisory updater.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323032	**Publication Service <%1> binds rvcm advisory updater.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323033	**Alerter configuration SQL: %1.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323034	**Alerter clean up SQL: %1.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323035	**Publication Service <%1> thread <%2> build fault status updater.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323036	**Publication Service <%1> thread <%2> bind fault status updater.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323037	**Publication Service <%1> thread <%2> build 'S' to 'N' updater.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323038	**Publication Service <%1> thread <%2> bind 'S' to 'N' updater.**		
	debugRole	Publication Service	Normal operation; no action is necessary.

Status Code	Role	Category	Resolution (Cont’d)
AEADB-323039	**Publication Service <%1> thread <%2> acquire table lock in load balance mode.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323040	**Publication Service <%1> thread <%2> release table lock in load balance mode.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323041	**Publication Service <%1> thread <%2> build child selector.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-323042	**Publication Service <%1> thread <%2> bind child selector.**		
	debugRole	Publication Service	Normal operation; no action is necessary.
AEADB-330001	**Publication Service <%1> throw exception <%2> when create rvcm batch confirm timer.**		
	errorRole	Publication	Check the Publication Service configuration and data in the publishing table.
AEADB-330002	**Publication Service <%1> throw exception <%2> when activate rvcm batch confirm timer.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330003	**Publication Service <%1> thread <%2> throw exception %3 when get table semaphore in multithreads mode.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330004	**Publication Service <%1> thread <%2> throw exception %3 when stop the adapter.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.

Status Code	Role	Category	Resolution (Cont’d)
AEADB-330005	**Publication Service <%1> throw exception <%2> when create polling timer.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330006	**Publication Service <%1> thread <%2> throw exception %3 when build pub load balancing marker.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330007	**Publication Service <%1> thread <%2> throw exception %3 when build pub load balancing pub table selector.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330008	**Publication Service <%1> thread <%2> throw exception %3 when build publishing table mini sequence selector.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330009	**Publication Service <%1> thread <%2> throw exception %3 when execute the mini sequence selector.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330010	**Publication Service <%1> thread <%2> throw exception %3 when bind pub load balancing marker.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330011	**Publication Service <%1> thread <%2> throw exception %3 when execute pub load balancing marker.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.

Status Code	Role	Category	Resolution (Cont'd)
AEADB-330012	**Publication Service <%1> thread <%2> throw exception %3 when build publishing table selector.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330013	**Publication Service <%1> thread <%2> throw exception %3 when bind publishing table selector.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330014	**Publication Service <%1> thread <%2> throw exception %3 when execute publishing table selector.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330015	**Publication Service <%1> thread <%2> throw exception %3 when get dbTable whose name is publishingTableName.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330016	**Publication Service <%1> thread <%2> throw exception %3 when construct MInstance.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330017	**Publication Service <%1> thread <%2> throw exception %3 when build reference object selector.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330018	**Publication Service <%1> thread <%2> throw exception %3 when bind reference object selector.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.

Status Code	Role	Category	Resolution (Cont'd)
AEADB-330019	**Publication Service <%1> thread <%2> throw exception %3 when construct minstance based on publishing table.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330020	**Publication Service <%1> thread <%2> throw exception %3 when construct minstance based on source table.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330021	**Publication Service <%1> thread <%2> throw exception %3 when construct minstance based on reference object table.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330022	**Publication Service <%1> thread <%2> throw exception %3 when fetch child data.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330023	**Publication Service <%1> thread <%2> throw exception %3 when construct publisher endpoint.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330024	**Publication Service <%1> thread <%2> throw exception %3 when publish group message.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330025	**Publication Service <%1> thread <%2> throw exception %3 when build pub table updater.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.

Status Code	Role	Category	Resolution (Cont'd)
AEADB-330026	**Publication Service <%1> thread <%2> throw exception %3 when bind pub table updater.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330027	**Publication Service <%1> thread <%2> throw exception %3 when update entry status.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330028	**Publication Service <%1> throw exception %2 when build rvcm advisory updater.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330029	**Publication Service <%1> throw exception %2 when bind rvcm advisory updater.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330030	**Publication Service <%1> throw exception %2 when deactivate the RVCM batch confirm timer.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330031	**Publication Service <%1> throw exception %2 when reset rvcm batch confirm timer.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330032	**Main thread throw exception %1 when configure alerter of %2. SQLSTRING : %3.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.

Status Code	Role	Category	Resolution (Cont'd)
AEADB-330033	**Main thread throw exception %1 when build listen_alert statement for alerter. SQL String : %2.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330034	**Alerter thread throw exception %1 when execute listen_alert statement.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330035	**Alerter thread throw exception %1 when clean up alerter. SQL String : %2.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330036	**Publication Service <%1> thread <%2> throw exception %3 when manipulate fault status updater.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330037	**Publication Service <%1> thread <%2> throw exception %3 when publish single message.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330038	**Publication Service <%1> thread <%2> throw exception %3 when manipulate S to N status updater.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330039	**Publication Service <%1> thread <%2> failed to construct minstance based on source table because no data fetched from %3.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330040	**Publication Service <%1> thread <%2> failed to construct minstance based on reference object because no data fetched from %3.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.

Status Code	Role	Category	Resolution (Cont'd)
AEADB-330041	**Publication Service <%1> thread <%2> will cause adapter terminate because the Exception: %3.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330042	**Publication Service <%1> thread <%2> throw exception %3 when build child selector.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330043	**Publication Service <%1> thread <%2> throw exception %3 when bind child selector.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330044	**Publication Service <%1> thread <%2> throw exception %3 when initialize the DBLockMechanism.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-330045	**Publication Service <%1> throw exception %2 when acquire the table lock.**		
	errorRole	Publication Service	Check the Publication Service configuration and data in the publishing table.
AEADB-400001	**Subscription Service <%1> receive one message.**		
	infoRole	Subscription Service	Normal operation; no action is necessary.
AEADB-400002	**Subscription Service Thread <%1> receive a group message which contain %2 record(s).**		
	infoRole	Subscription Service	Normal operation; no action is necessary.
AEADB-400003	**Subscription Service Thread <%1> receive a single message.**		
	infoRole	Subscription Service	Normal operation; no action is necessary.

Status Code	Role	Category	Resolution (Cont’d)
AEADB-400004	**Subscription Service Thread <%1> start to handle the %2 item for the group message.**		
	infoRole	Subscription Service	Normal operation; no action is necessary.
AEADB-400005	**Subscription Service Thread <%1> batch commit timeout.**		
	infoRole	Subscription Service	Normal operation; no action is necessary.
AEADB-400006	**Subscription Service Thread <%1> %2 row(s) affected.**		
	infoRole	Subscription Service	Normal operation; no action is necessary.
AEADB-400007	**Subscription Service Thread <%1> send reply \n%2\n to destination %3.**		
	infoRole	Subscription Service	Normal operation; no action is necessary.
AEADB-410001	**Subscription Service Thread <%1> discard one message.**		
	warnRole	Subscription Service	Normal operation; no action is necessary.
AEADB-421001	**Subscription Service Thread <%1> start DB transaction.**		
	debugRole	Subscription Service	Normal operation; no action is necessary.
AEADB-421002	**Subscription Service Thread <%1> commit DB transaction.**		
	debugRole	Subscription Service	Normal operation; no action is necessary.
AEADB-421003	**Subscription Service Thread <%1> roll back DB transaction.**		
	debugRole	Subscription Service	Normal operation; no action is necessary.
AEADB-421004	**Subscription Service Thread <%1> bypass one message.**		
	debugRole	Subscription Service	Normal operation; no action is necessary.

Status Code	Role	Category	Resolution (Cont'd)
AEADB-421005	**Subscription Service Thread <%1> the statement in cache cannot be reused .**		
	debugRole	Subscription Service	Normal operation; no action is necessary.
AEADB-421006	**Subscription Service Thread <%1> call pre-commit procedure successfully.**		
	debugRole	Subscription Service	Normal operation; no action is necessary.
AEADB-421007	**Subscription Service Thread <%1> create statement.**		
	debugRole	Subscription Service	Normal operation; no action is necessary.
AEADB-421008	**Subscription Service Thread <%1> bind statement.**		
	debugRole	Subscription Service	Normal operation; no action is necessary.
AEADB-421009	**Subscription Service Thread <%1> execute statement.**		
	debugRole	Subscription Service	Normal operation; no action is necessary.
AEADB-421010	**Subscription Service Thread <%1> flush Bulk-Insert.**		
	debugRole	Subscription Service	Normal operation; no action is necessary.
AEADB_422001	**Subscription Service Thread <%1> Confirms %2 error message Success.**		
	debugRole	Subscription Service	Check the message data or the table to make sure whether you need this message to throw the exception.
AEADB_422002	**Subscription Service Thread <%1> Confirming %2 message Successfully.**		
	debugRole	Subscription Service	Normal operation; no action is necessary.
AEADB_422003	**Subscription Service Thread <%1> statement SQL: %2.**		
	debugRole	Subscription Service	Normal operation; no action is necessary.

Status Code	Role	Category	Resolution (Cont'd)
AEADB_422004	**Subscription Service Thread <%1> will do incremental operation for child records.**		
	debugRole	Subscription Service	Normal operation; no action is necessary.
AEADB_422005	**Subscription Service Thread <%1> will do completion operation for child records.**		
	debugRole	Subscription Service	Normal operation; no action is necessary.
AEADB_422006	**Subscription Service Thread <%1> insert the data %2 into opaque exception table %3.**		
	debugRole	Subscription Service	Normal operation; no action is necessary.
AEADB_422007	**Subscription Service Thread <%1> insert the data %2 into exception table %3.**		
	debugRole	Subscription Service	Normal operation; no action is necessary.
AEADB-423001	**Subscription Service Thread <%1> result of pre-commit stored procedure %2 is %3.**		
	debugRole	Subscription Service	Normal operation; no action is necessary.
AEADB-423002	**Subscription Service Thread <%1> statement bounds: %2.**		
	debugRole	Subscription Service	Normal operation; no action is necessary.
AEADB-423003	**Subscription Service Thread <%1> query related child records %2.**		
	debugRole	Subscription Service	Normal operation; no action is necessary.
AEADB-423004	**Subscription Service Thread <%1> delete related child records %2.**		
	debugRole	Subscription Service	Normal operation; no action is necessary.

Status Code	Role	Category	Resolution (Cont'd)
AEADB-423005	**Subscription Service Thread <%1> cache statement %2 for table %3.**		
	debugRole	Subscription Service	Normal operation; no action is necessary.
AEADB-423006	**Subscription Service Thread <%1> Message body: %2.**		
	debugRole	Subscription Service	Normal operation; no action is necessary.
AEADB-423007	**Subscription Service Thread <%1> deactivate batch commit timer.**		
	debugRole	Subscription Service	Normal operation; no action is necessary.
AEADB-423008	**Subscription Service Thread <%1> activate batch commit timer.**		
	debugRole	Subscription Service	Normal operation; no action is necessary.
AEADB-423009	**Subscription Service <%1>, Message queue ID calculation time: %2.**		
	debugRole	Subscription Service	Normal operation; no action is necessary.
AEADB-423010	**Subscription Service <%1>, Dispatching the message to Queue ID: %2.**		
	debugRole	Subscription Service	Normal operation; no action is necessary.
AEADB-423011	**Subscription Service <%1> thread <%2> set empty string for column <%3>.**		
	debugRole	Subscription Service	Normal operation; no action is necessary.
AEADB-430001	**Subscription Service <%1> received unexpected data %2.**		
	errorRole	Subscription Service	Check the message data.
AEADB-430002	**Subscription Service <%1> throw exception %2 when parse data %3.**		
	errorRole	Subscription Service	Check the message data.

Status Code	Role	Category	Resolution (Cont'd)
AEADB-430003	**Subscription Service <%1> throw exception %2 when calculate queueID for data %3.**		
	errorRole	Subscription Service	Check the message data.
AEADB-430004	**Subscription Service Thread <%1> throw exception %2 when check if the %3 is a kind of group message.**		
	errorRole	Subscription Service	Check the message data.
AEADB-430005	**Subscription Service Thread <%1> throw exception %2 when create statement for Data %3.**		
	errorRole	Subscription Service	Check the message data or the table to make sure whether you need this message to throw the exception.
AEADB-430006	**Subscription Service Thread <%1> throw exception %2 when terminate adapter.**		
	errorRole	Subscription Service	Check the logs and the configuration.
AEADB-430007	**Subscription Service Thread <%1> throw exception %2 when bind parameter to statement %3.**		
	errorRole	Subscription Service	Check the message data or the table to make sure whether you need this message to throw the exception.
AEADB-430008	**Subscription Service Thread <%1> throw exception %2 when execute statement %3.**		
	errorRole	Subscription Service	Check the message data or the table to make sure whether you need this message to throw the exception.
AEADB-430009	**Subscription Service Thread <%1> throw exception %2 when create batch commit timer %3.**		
	errorRole	Subscription Service	Check the message data or the table to make sure whether you need this message to throw the exception.

Status Code	Role	Category	Resolution (Cont'd)
AEADB-430010	**Subscription Service Thread <%1> throw exception %2 when handle timeout event.**		
	errorRole	Subscription Service	Check the message data or the table to make sure whether you need this message to throw the exception.
AEADB-430011	**Subscription Service Thread <%1> call pre-commit procedure unsuccessfully.**		
	errorRole	Subscription Service	Check the message data or the table to make sure whether you need this message to throw the exception.
AEADB-430012	**Subscription Service Thread <%1> throw exception %2 when serialize the message %3.**		
	errorRole	Subscription Service	Check the message data or the table to make sure whether you need this message to throw the exception.
AEADB-430013	**Subscription Service Thread <%1> throw exception %2 when activate batch commit timer %2.**		
	errorRole	Subscription Service	Check the message data or the table to make sure whether you need this message to throw the exception.
AEADB-430014	**Subscription Service Thread <%1> throw exception %2 when deactivate batch commit timer %2.**		
	errorRole	Subscription Service	Check the message data or the table to make sure whether you need this message to throw the exception.
AEADB-430015	**Subscription Service Thread <%1> handle [%2] : %3.**		
	errorRole	Subscription Service	Check the message data or the table to make sure whether you need this message to throw the exception.
AEADB-430016	**Subscription Service Thread <%1> the exception table is not found.**		
	errorRole	Subscription Service	Check if you set the exception table.

Status Code	Role	Category	Resolution (Cont'd)
AEADB-430017	**Subscription Service Thread <%1> the opaque exception table is not found.**		
	errorRole	Subscription Service	Check if you set the opaque exception table.
AEADB-430018	**Subscription Service Thread <%1> the child exception table %2 is not found.**		
	errorRole	Subscription Service	Check if you set the child exception table.
AEADB-430019	**Subscription Service Thread <%1> throw exception %2 when begin transaction.**		
	errorRole	Subscription Service	Check the database.
AEADB-430020	**Subscription Service Thread <%1> send reply %2 to destination %3 Error data: %4.**		
	errorRole	Subscription Service	Check the message data.
AEADB-430021	**Subscription Service Thread <%1> throw exception %2 when roll back transaction.**		
	errorRole	Subscription Service	Check the database.
AEADB-430022	**Subscription Service Thread <%1> throw exception %2 when set custom scale.**		
	errorRole	Subscription Service	Check the custom scale.
AEADB-500001	**RPC Service Thread <%1> receive data.**		
	infoRole	RPC	Normal operation; no action is necessary
AEADB-500002	**RPC Service Thread <%1> reply message to %2.**		
	infoRole	RPC	Normal operation; no action is necessary
AEADB-500003	**RPC Service Thread <%1> Database operated successful.**		
	infoRole	RPC	Normal operation; no action is necessary

Status Code	Role	Category	Resolution (Cont'd)
AEADB-500004	**RPC Service Thread <%1> receive one message.**		
	infoRole	RPC	Normal operation; no action is necessary
AEADB-500005	**RPC Service Thread <%1> Reply Message body is: %2.**		
	infoRole	RPC	Normal operation; no action is necessary
AEADB-521001	**RPC Service Thread <%1> begin transaction.**		
	debugRole	RPC	Normal operation; no action is necessary
AEADB-521002	**RPC Service Thread <%1> commit transaction.**		
	debugRole	RPC	Normal operation; no action is necessary
AEADB-521003	**RPC Service Thread <%1> roll back transaction.**		
	debugRole	RPC	Check the message data and the operation logs.
AEADB-521004	**RPC Service Thread <%1> execute statement.**		
	debugRole	RPC	Normal operation; no action is necessary
AEADB-521005	**RPC Service Thread <%1> create statement.**		
	debugRole	RPC	Normal operation; no action is necessary
AEADB-522001	**RPC Service Thread <%1> statement SQL: %2.**		
	debugRole	RPC	Normal operation; no action is necessary
AEADB-522002	**RPC Service Thread <%1> Confirming message Success.**		
	debugRole	RPC	Normal operation; no action is necessary
AEADB-523001	**Request-Response (Custom mode) Service <%1> initialize Operation <%2>.**		
	debugRole	RPC	Normal operation; no action is necessary
AEADB-523002	**RPC Service Thread <%1> statement binding data: %2.**		
	debugRole	RPC	Normal operation; no action is necessary
AEADB-523003	**<%1> Operation: %2.**		

Status Code	Role	Category	Resolution (Cont'd)
	debugRole	RPC	Normal operation; no action is necessary
AEADB-523004	**RPC Service Thread <%1> do not found reply subject in the dateEvent.**		
	debugRole	RPC	Check the configuration.
AEADB-523005	**RPC Service Thread <%1> Statement Cached: %2.**		
	debugRole	RPC	Normal operation; no action is necessary.
AEADB-523006	**RPC Service Thread <%1> Receive Message body is: %2.**		
	debugRole	RPC	Normal operation; no action is necessary.
AEADB-530001	**RPC Service Thread <%1> throw exception: %2 when create DB Statement for Data: %3.**		
	errorRole	RPC	Check the message data and the operation logs.
AEADB-530002	**RPC Service Thread <%1> throw exception: %2 when begin transaction.**		
	errorRole	RPC	Check the message data and the operation logs.
AEADB-530003	**RPC Service Thread <%1> throw exception: %2 when execute DB Statement %3.**		
	errorRole	RPC	Check the message data and the operation logs.
AEADB-530004	**RPC Service Thread <%1> throw exception: %2 when commit transaction.**		
	errorRole	RPC	Check the message data and the operation logs.
AEADB-530005	**RPC Service Thread <%1> throw exception: %2 when roll back transaction.**		
	errorRole	RPC	Check the message data and the operation logs.
AEADB-530006	**RPC Service Thread <%1> throw exception: %2 when reply message to %3.**		
	errorRole	RPC	Check the message data and the operation logs.
AEADB-530007	**RPC Service Thread <%1> Database operated unsuccessful.**		
	errorRole	RPC	Check the message data and the operation logs.
AEADB-530008	**RPC Service Thread <%1> throw exception: %2 when get DB procedure %3.**		

Status Code	Role	Category	Resolution (Cont'd)
	errorRole	RPC	Check the message data and the operation logs.
AEADB-530009	**RPC Service Thread <%1> throw exception: %2 create operation %3.**		
	errorRole	RPC	Check the message data and the operation logs.
AEADB-530010	**RPC Service Thread <%1> throw exception: %2 when parse request %3.**		
	errorRole	RPC	Check the message data and the operation logs.
AEADB-530011	**Subscription Service <%1> received unexpected data %2.**		
	errorRole	RPC	Check the message data and the operation logs.
AEADB-600001	**Hawk Method %1 is invoked.**		
	infoRole	Hawk	Normal operation; no action is necessary.
AEADB-822001	**%1 reconnect attempt %2.**		
	debugRole	Database	Check the database connection.
AEADB-823001	**%1 reconnect failed on attempt %2.**		
	debugRole	Database	Check the database connection.
AEADB-823002	**Reconnection thread will sleep %1ms.**		
	debugRole	Database	Check the database connection.
AEADB-823003	**Oracle client info set successfully in the connection of <%1>...**		
	debugRole	Database	Normal operation; no action is necessary.
AEADB-921001	**Advisory Handler receives RVCM Advisory on %1.**		
	debugRole	Advisory	Normal operation; no action is necessary.
AEADB-923001	**Advisory Handler looks for the matched Publication Service by subject %1.**		
	debugRole	Advisory	Normal operation; no action is necessary.
AEADB-923002	**Advisory Handler finds the matched Publication Service %1.**		
	debugRole	Advisory	Normal operation; no action is necessary.

Status Code	Role	Category	Resolution (Cont'd)
AEADB-923003	**Advisory Handler does not find the matched Publication Service.**		
	debugRole	Advisory	Normal operation; no action is necessary.
AEADB-923004	**Received RV SYSTEM Advisory on %1 %2.**		
	debugRole	Advisory	Normal operation; no action is necessary.
AEADB-930001	**Received SDK error Advisory on %1 %2.**		
	errorRole	Advisory	SDK error. Check the logs and the configuration.
AEADB-022001	**Performance Status: %1.**		
	debugRole	Performance	Normal operation; no action is necessary.
AEADB-024001	**Exception: %1.**		
	debugRole	Exception	Check the logs and the configuration.
AEADB-030001	**Got SDK error in method <%1>: Error code: %2 Error Message: %3.**		
	errorRole	Exception	SDK error. Check the logs and the configuration.
AEADB-030002	**Refresh performance logging failed, please enable the performance logging when start the adapter.**		
	errorRole	Performance	Check the logs and the configuration.