TIBCO ActiveMatrix® Adapter for Tuxedo

Examples

Software Release 6.0
March 2010

Important Information

Contents

Figures . . . . . **vii**

Tables . . . . . **ix**

Preface . . . . . **xi**

Related Documentation xii
- TIBCO ActiveMatrix Adapter for Tuxedo Documentation xii
- Other TIBCO Product Documentation xii
- Third-Party Documentation xiii

Typographical Conventions xiv

How to Contact TIBCO Support xvii

Chapter 1 Introduction **1**

Overview 2
- Using TIBCO ActiveMatrix BusinessWorks 3
- Working with dat files in TIBCO Designer 4
- Location and Directory Structure 5

Data Flow 7
- Data Flow for Publication Service Examples 7
- Data Flow for Request-Response Service Examples 8
- Data Flow for Subscription Service Examples 9
- Data Flow for Request-Response Invocation Service Examples 10

Chapter 2 Publication Service for Rendezvous Adapter Agent-Based Communication **13**

Example Description 14
- Location of the Example Files 14

Setup the Example 16

Deploy and Run the Example 17

Expected Results 19

Chapter 3 Publication Service for JMS Adapter Agent-Based Communication **21**

Example Description 22
- Location of the Example Files 22

Setup the Example 24

Run and Test the Example 25
Expected Results 26

Chapter 4 Publication Service for Tuxedo Queue-Based Communication 27

Example Description 28
Location of the Example Files 28
Setup the Example 30
Run and Test the Example 31
Expected Results 32

Chapter 5 Subscription Service for Asynchronous Communication 33

Example Description 34
Location of the Example Files 35
Setup the Example 37
Run and Test the Example 38
Expected Results 40

Chapter 6 Subscription Service for Conversational Communication 41

Example Description 42
Location of the Example Files 43
Setup the Example 45
Run and Test the Example 46
Expected Results 48

Chapter 7 Subscription Service for Queue-Based Communication 51

Example Description 52
Location of the Example Files 53
Setup the Example 55
Run and Test the Example 56
Expected Results 58

Chapter 8 Request-Response Service for Synchronous Communication 59

Example Description 60
Location of the Example Files 61
Setup the Example 63
Run and Test the Example 64
Expected Results 66

Chapter 9 Request-Response Service for Asynchronous Communication 67

Example Description 68
Location of the Example Files 68
Setup the Example 70
Run and Test the Example 71
Expected Results 73

Chapter 10 Request-Response Invocation Service for Rendezvous Adapter Agent-Based Communication 75

Example Description 76
Location of the Example Files 76
Setup the Example 78
Run and Test the Example 79

Chapter 11 TIBCO ActiveMatrix BusinessWorks: End-to-End Publish-Subscribe for Event Based Communication 81

Example Description 82
Location of the Example Files 82
Setup the Example 84
Run and Test the Example 85
Expected Results 87

Appendix A Tasks for Preparing Tuxedo 91

Task A: Edit ubbconfig File 92
Task B: Edit the Make Script 93
Task C: Run the Make Script 94

Index 95

Figures

Figure 1 Examples Directory Structure. 6
Figure 2 Flow of Events for the Publication Service Example 7
Figure 3 Flow of Events for the Request-Response Service Example. 8
Figure 4 Flow of Events for the Subscription Service Example 9
Figure 5 Flow of Events for the Request-Response Invocation Service Example 10

Tables

Table 1 General Typographical Conventions xiv
Table 2 Syntax Typographical Conventions xv
Table 3: Examples Provided with the Adapter Installation 2

Preface

TIBCO ActiveMatrix Adapter for Tuxedo ships with pre configured examples. This manual describes how to run these examples, and explains the configuration of each.

Topics

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

Related Documentation

This section lists documentation resources you may find useful.

TIBCO ActiveMatrix Adapter for Tuxedo Documentation

The following documents form the TIBCO ActiveMatrix Adapter for Tuxedo documentation set:

- *TIBCO ActiveMatrix Adapter for Tuxedo Concepts* Read this manual to familiarize yourself with the concepts used by this product.
- *TIBCO ActiveMatrix Adapter for Tuxedo Installation* Read this manual to learn how to install TIBCO ActiveMatrix Adapter for Tuxedo.
- *TIBCO ActiveMatrix Adapter for Tuxedo Configuration and Deployment* Read this manual for instructions on creating, configuring , and deploying adapter projects.
- *TIBCO ActiveMatrix Adapter for Tuxedo Examples* Read this manual to work through the examples provided with the adapter.
- *TIBCO ActiveMatrix Adapter for Tuxedo Release Notes* Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.

Other TIBCO Product Documentation

You may find it useful to read the documentation for the following TIBCO products. Note that only books that relate to adapters are listed. Each of the books is available from the `doc` directory in the product's installation area.

- TIBCO Designer™
- TIBCO Administrator™
- TIBCO ActiveMatrix BusinessWorks™
- TIBCO Rendezvous®
- TIBCO Enterprise Message Service™
- TIBCO Hawk®
- TIBCO Adapter™ SDK
- TIBCO Runtime Agent™

Third-Party Documentation

You may also find it useful to read the Oracle Tuxedo User documentation.

Typographical Conventions

The following typographical conventions are used in this manual.

Table 1 General Typographical Conventions

Convention	Use
TIBCO_HOME *ENV_HOME*	Many TIBCO products must be installed within the same home directory. This directory is referenced in documentation as *TIBCO_HOME*. The value of *TIBCO_HOME* depends on the operating system. For example, on Windows systems, the default value is `C:\tibco`. Other TIBCO products are installed into an installation environment. Incompatible products and multiple instances of the same product are installed into different installation environments. The directory into which such products are installed is referenced in documentation as *ENV_HOME*. The value of *ENV_HOME* depends on the operating system. For example, on Windows systems the default value is C:\tibco.
`code font`	Code font identifies commands, code examples, filenames, pathnames, and output displayed in a command window. For example: Use `MyCommand` to start the foo process.
`bold code font`	Bold code font is used in the following ways: • In procedures, to indicate what a user types. For example: Type **`admin`**. • In large code samples, to indicate the parts of the sample that are of particular interest. • In command syntax, to indicate the default parameter for a command. For example, if no parameter is specified, `MyCommand` is enabled: `MyCommand [`**`enable`**` | disable]`
italic font	Italic font is used in the following ways: • To indicate a document title. For example: See *TIBCO ActiveMatrix BusinessWorks Concepts*. • To introduce new terms For example: A portal page may contain several portlets. *Portlets* are mini-applications that run in a portal. • To indicate a variable in a command or code syntax that you must replace. For example: `MyCommand` *PathName*

Table 1 General Typographical Conventions (Cont?)

Convention	Use
Key combinations	Key name separated by a plus sign indicate keys pressed simultaneously. For example: Ctrl+C. Key names separated by a comma and space indicate keys pressed one after the other. For example: Esc, Ctrl+Q.
	The note icon indicates information that is of special interest or importance, for example, an additional action required only in certain circumstances.
	The tip icon indicates an idea that could be useful, for example, a way to apply the information provided in the current section to achieve a specific result.
	The warning icon indicates the potential for a damaging situation, for example, data loss or corruption if certain steps are taken or not taken.

Table 2 Syntax Typographical Conventions

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

Table 2 Syntax Typographical Conventions

Convention	Use

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

How to Contact TIBCO Support

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

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

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

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

  https://support.tibco.com

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

Chapter 1 Introduction

This chapter contains an overview of the examples that are packaged with the adapter. The examples are explained in the later chapters of this guide.

Topics

- *Overview, page 2*
- *Data Flow, page 7*

Overview

The following examples demonstrate the integration between the adapter and TIBCO ActiveMatrix BusinessWorks. The different communication paradigms and common business objects have been chosen to build an end-to-end enterprise wide integration and demonstrate the adapter capabilities.

The following table shows the communication paradigm, business object and the corresponding adapter service used to develop the examples.

Table 3: Examples Provided with the Adapter Installation

Supported Communication Paradigm	Adapter Service	Business Object
Event-Based Communication	Publish-Subscribe	Employee
Rendezvous Adapter Agent-Based Communication	Publication	PurchaseOrder
JMS Adapter Agent-Based Communication	Publication	JMSTopicPub
Queue-based Communication	Publication	PurchaseOrder
Asynchronous Communication	Subscription	Customer
Conversational Communication	Subscription	Material
Queue-based Communication	Subscription	Employee
Synchronous Communication	Request-Response	Employee
Asynchronous Communication	Request-Response	Product
Agent-Based Communication	Request-Response Invocation	SalesOrder

- The TIBCO ActiveMatrix BusinessWorks examples use both the transport types, RV and JMS. The procedure to arrive at the configurations using TIBCO Rendezvous is similar to that of configurations using JMS.
- All examples use `FML32` as buffer type.
- All examples provided are for the adapter in workstation client mode. (To work with these examples in the native mode, `WSNADDR` should not be set. The `ubbconfig` file has to be modified so that the `WSNADDR` section should not be present)

- Some examples obtain input data from and write output data into files. And some examples use the client to send data.

Before running the examples, ensure that you change the paths of the input and output XMLs to the appropriate locations according to your *TIBCO_HOME* installation.

- The example repositories have embedded TIBCO ActiveMatrix BusinessWorks processes.
- Following is provided as part of each example:
  - A sample Tuxedo service source code
  - The Tuxedo FML32 header files
  - `ubbconfig` files
  - Batch files to compile and boot the Tuxedo service
  - Sample project zip file for the TIBCO ActiveMatrix BusinessWorks examples

Using TIBCO ActiveMatrix BusinessWorks

If you are using the adapter with TIBCO ActiveMatrix BusinessWorks, the following software must be installed to run the examples. For the supported versions of the listed products, refer to *TIBCO ActiveMatrix Adapter for Tuxedo readme.txt*.

- TIBCO ActiveMatrix BusinessWorks
- TIBCO Administrator
- TIBCO ActiveMatrix Adapter for Tuxedo
- Tuxedo (For a list of supported Tuxedo versions, refer to *TIBCO ActiveMatrix Adapter for Tuxedo Installation*)
- TIBCO Runtime Agent
- TIBCO Enterprise Message Service

TIBCO Enterprise Message Service must be installed to run examples that use JMS as the transport. The EMS server must be running and accessible to the machine on which the adapter is installed.

The TIBCO ActiveMatrix BusinessWorks examples use TIBCO Designer to create an Enterprise Archive File (EAR). One of the examples explains deploying the EAR file using TIBCO Administrator Enterprise Edition. All the other examples use the Adapter tester functionality of TIBCO Designer to run and test the examples.

In TIBCO Administrator, make sure all software components needed by the adapter instance are installed on one or more machines that are part of a TIBCO Administration Domain and that the software is registered in the domain.

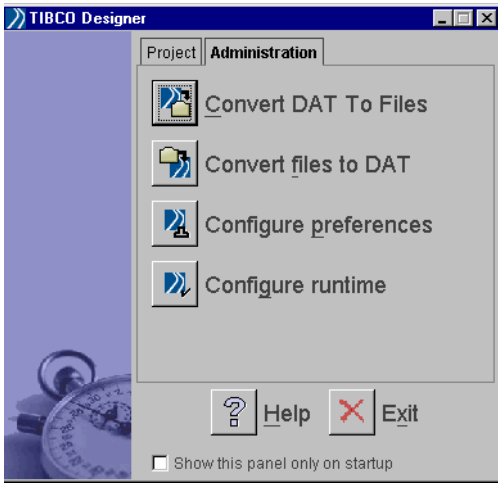
- Use the TIBCO Domain Utility to add a machine to a TIBCO Administration Domain.

These topics are explained in the TIBCO Administrator documentation set.

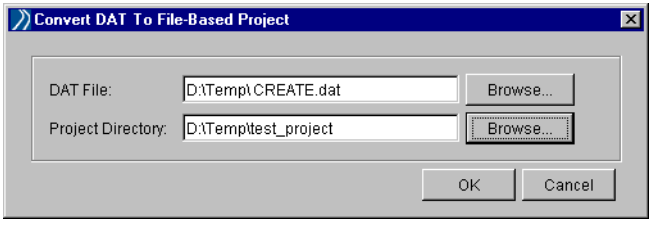
Working with dat files in TIBCO Designer

You cannot directly open a dat file in TIBCO Designer and make modifications to the configurations. To do this, complete the following steps:

1. Convert the dat file to a multi-file project.
   a. Open TIBCO Designer. In the first screen that is displayed, click **Administration**. The screen is shown next.



   b. Then click the **Convert DAT to Files** icon. In the window that is displayed, browse and select the dat file you wish to convert to a multi-file project. Click **OK**.



2. Click the **Open existing project** icon from the first TIBCO Designer screen. Browse to the directory where the converted multi-file project is saved.
3. Make configuration changes as per your requirements.

Export the multi-file project to a dat. Select **Project>Export Full Project** from the menu. Browse and select the location of the directory you wish to save the dat file to. Ensure that the directory is different from the multi-file project. Enter the name of the project and click **OK**.

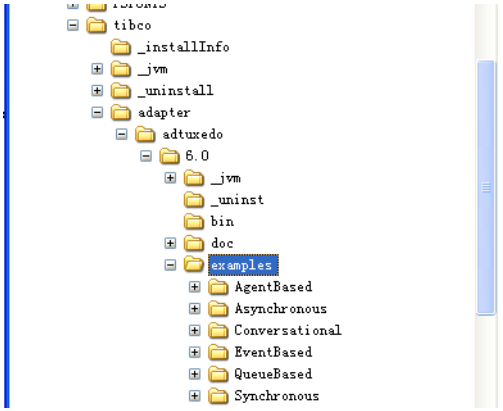
Location and Directory Structure

The samples are available in the following location of the adapter installation.

TIBCO_HOME`\adapter\adtuxedo\`*version_number*`\examples`

The directory structure is shown next.

Figure 1 Examples Directory Structure



For better organization the examples have been supplied in the given structure. Tuxedo imposes a limit on the length of characters for the variables that are set in the `ubbconfig` file. So before running any example, make a copy of the complete examples directory to a different location, which meets Tuxedo's character length limits, modify as required and run them from that location.

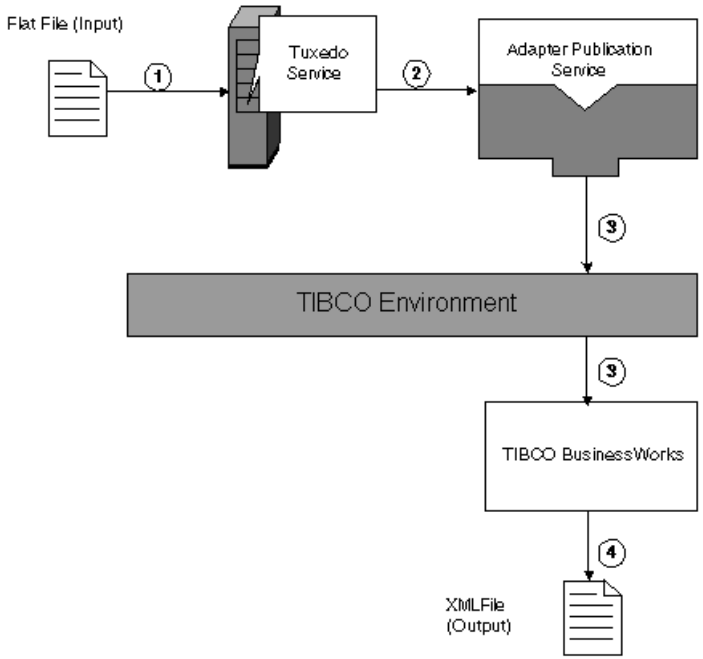
Data Flow

This section describes the flow of events for all the examples provided with the adapter installation.

Data Flow for Publication Service Examples

A diagrammatic representation of the data flow for the Publication service examples is given next.

Figure 2 Flow of Events for the Publication Service Example



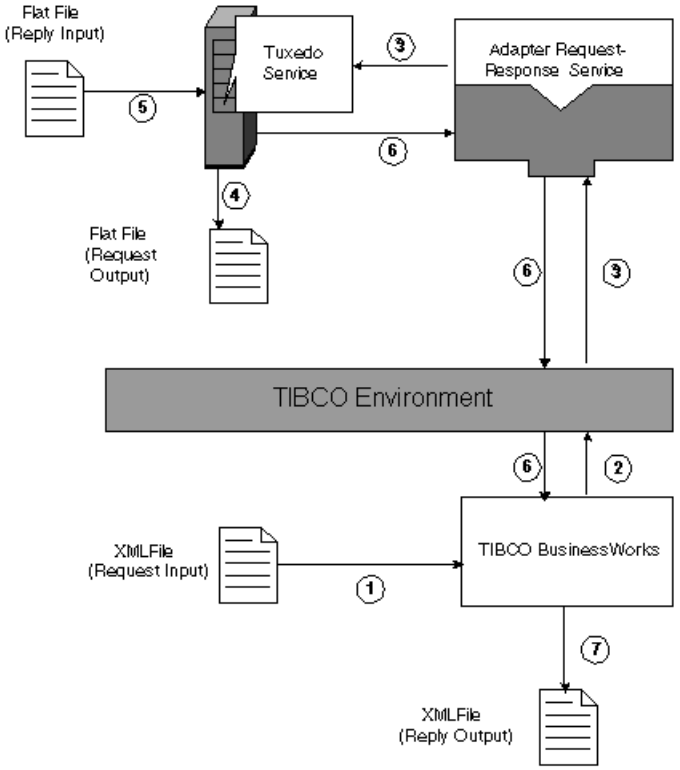
1. The Tuxedo service reads the data from an input flat file.
2. The service sends the data to the adapter.
   - In the Event based example, the Tuxedo service posts the data onto the EventBroker.
   - In the Rendezvous Adapter Agent-based example, the agent sends the data to the adapter, using TIBCO Rendezvous (RV) transport type.
   - In the JMS Adapter Agent-based example, the agent sends the data onto JMS server, from which the adapter receives the data.

3. The adapter publishes the data and the subscriber in TIBCO ActiveMatrix BusinessWorks subscribes to the data.
4. The TIBCO ActiveMatrix BusinessWorks process writes the data to an output XML file.

Data Flow for Request-Response Service Examples

A diagrammatic representation of the data flow for the Request-Response service examples is given next.

Figure 3 Flow of Events for the Request-Response Service Example



1. The TIBCO ActiveMatrix BusinessWorks process reads the data from an XML file.
2. The client configured in TIBCO ActiveMatrix BusinessWorksrequests the data and the adapter gets the request.
3. The adapter in turn invokes a Tuxedo Service, which reads from a file and returns the reply to TIBCO ActiveMatrix BusinessWorks.

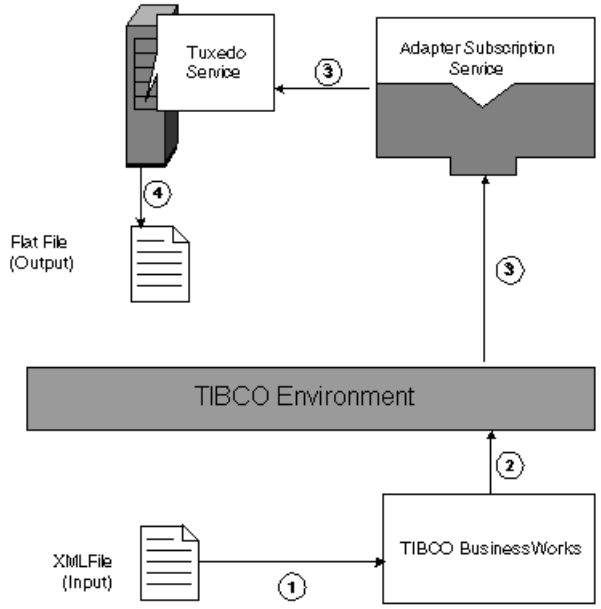
4. The Tuxedo Service writes the request to a flat file.
5. The Tuxedo service then reads from a flat file.
6. It sends a response to the adapter, which forwards it to TIBCO ActiveMatrix BusinessWorks.
7. The TIBCO ActiveMatrix BusinessWorks process then writes the data to an XML file.

Subscription service operating in Conversational and Queue paradigms are similar to Request-Response service since they send a reply on a reply subject.

Data Flow for Subscription Service Examples

A diagrammatic representation of the data flow for the Subscription service examples is given next.

Figure 4 Flow of Events for the Subscription Service Example



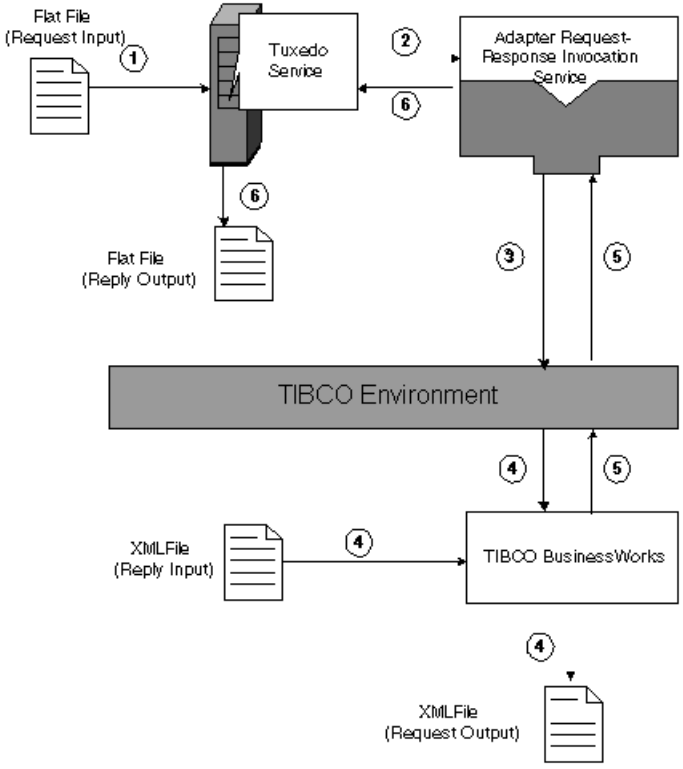
1. The TIBCO ActiveMatrix BusinessWorks process reads the data from an XML file.
2. The publisher configured in TIBCO ActiveMatrix BusinessWorks publishes the data and the adapter subscription service subscribes to the data.
3. The adapter Subscription service in turn invokes a Tuxedo Service.

4. The Tuxedo service writes the data to a flat file.

Data Flow for Request-Response Invocation Service Examples

A diagrammatic representation of the data flow for the Request-Response Invocation service examples is given next.

Figure 5 Flow of Events for the Request-Response Invocation Service Example



1. The Tuxedo service reads the data from a flat file.
2. The service invokes the adapter Request-Response Invocation service.
3. The adapter sends the request.
4. TIBCO ActiveMatrix BusinessWorks receives the request and writes it to the Request output XML file.
5. TIBCO ActiveMatrix BusinessWorks then reads the reply input file and sends a response.

6. The adapter Request-Response Invocation service receives the response and forwards it to the Tuxedo service, which in turn writes it to an XML output file.

Chapter 2

Publication Service for Rendezvous Adapter Agent-Based Communication

This example shows how to use the Publication service for the Rendezvous Adapter Agent-based communication within a TIBCO ActiveMatrix BusinessWorks process. The example is deployed and run using the TIBCO Administrator GUI.

This example uses TIBCO Rendezvous (RV) as its transport type between the adapter agent and the adapter, and uses Java Messaging Service (JMS) as its transport type between the adapter and TIBCO environment.

Topics

- *Example Description, page 14*
- *Setup the Example, page 16*
- *Deploy and Run the Example, page 17*
- *Expected Results, page 19*

Example Description

This example shows how the adapter Publication service can be used in a TIBCO ActiveMatrix BusinessWorks process. The project configured for this example has a TIBCO ActiveMatrix BusinessWorks process which uses adapter Publication service which in turn uses Rendezvous Adapter Agent-based communication.

The business object that is used for this example is `PurchaseOrder`. A Tuxedo service, based on the PurchaseOrder ID that it receives from the invoking client, returns the `POTYPE`, `UOM`, `PODATE`, `CURRENCYTYPE`, `STATUS`, `MFGID`, `PRICE` corresponding to that PurchaseOrder ID. The adapter Publication service publishes this data.

Following are the flow of events:

1. User sends request from the Tuxedo client called `Clt`.
2. The client invokes the `POINFO` Tuxedo service.
3. The Tuxedo service sends a request to the adapter through the adapter agent.
4. The adapter agent posts the data to the adapter.
5. The adapter publishes the request over RVCM transport.
6. TIBCO ActiveMatrix BusinessWorks receives the request and writes data in an XML output file.

Location of the Example Files

Files provided with this example are given in the next table. The files are located in the following locations:

TIBCO_HOME\adapter\adtuxedo*version_number*\examples\AgentBased\PurchaseOrder

TIBCO_HOME\adapter\adtuxedo*version_number*\examples\AgentBased\PurchaseOrder\BusinessWorks

File Name	Purpose
`Get_input_renderXsd.xsd`	XSD file with XML schema.
`PODetails.h`	Tuxedo header file containing the FML32 definition.

File Name	Purpose
PurchaseOrder_Pub_Agent.zip	Project zip that contains the adapter Publication service configuration and the TIBCO ActiveMatrix BusinessWorks process. For details on dat files, see Working with dat files in TIBCO Designer on page 4.
adapteragent.h	This is required to compile the Tuxedo server with the adapter agent. This is a copy of *TIBCO_HOME*\adapter*version_number*\include\adapteragent.h
client.c	Source file for the Tuxedo client, which invokes the Tuxedo service, POINFO.
Get_output.xml	The output XML file.
inputdata.txt	File containing input data in XML format.
make.bat	Contains commands required to get the Tuxedo service running on Windows (Sets the required environment variables, creates executables, loads the ubbconfig file and boots the Tuxedo service)
make.sh	Contains commands required to get the Tuxedo service running on UNIX (Sets the required environment variables, creates executables, loads the ubbconfig file and boots the Tuxedo service). This script should be run in a k-shell.
server.c	Source file for Tuxedo Server containing the Tuxedo service, POINFO.
ubbagent	ubbconfig file

Setup the Example

In TIBCO Designer:

1. In the initial dialog box click **Open New Project** and specify a name for the project.
2. Click **Project>Import Full Project**.
3. Click the Zip Archive tab and browse to locate the `PurchaseOrder_Pub_Agent.zip` file. The file is located in the *TIBCO_HOME*`\adapter\adtuxedo\`*version_number*`\examples\AgentBased\PurchaseOrder\BusinessWorks` folder.
4. Save as a multi-file project.
5. Select the **Enterprise Archive Object**. Click Build Archive in the Configuration tab. This creates an EAR file for the instance `TuxedoAdapterConfiguration`.
6. Exit TIBCO Designer.

Starting the Tuxedo Service

Bring up the sample Tuxedo services given in *TIBCO_HOME*`\adapter\adtuxedo\`*version_number*`\examples\AgentBased\PurchaseOrder` folder, so that the adapter can communicate with them. See Tasks for Preparing Tuxedo on page 91 for details.

Deploy and Run the Example

Perform the following tasks to deploy and run the example.

Task A Start the Adapter in TIBCO Administrator

1. Click on the **Installed Software** link on the left panel. Check if TIBCO Adapter for Tuxedo 6.0 is registered. If the software is not registered, register by clicking the **Add Custom Software** button. Please refer to the TIBCO Administrator documentation for more details on adding custom software.
2. Go to **Application Management**. Click **New Application**. Upload the EAR file created in the previous section and click **OK**. On the following screen, clear the Quick Configure check box and then click **Save**.
3. Click **Configuration** in the newly created application. Click on the top level application name in the configuration view.
4. Expand the **Enterprise archive** link, so that adapter instance is visible.
5. Click on the *<InstanceName>.ear* and then click the **Add to Additional Machines** button.
6. Select the machine and click **OK**. In the following screen, click **Save**.
7. In the Configuration screen click **Deploy**, which will take you to the next screen. The Start successfully deployed services check box is selected by default. Click **OK**. The .tra files are created in `<Adapter_Home>/bin/domain/<DomainName>`.
8. Once the deployment is complete, click **Service Instances** under the application. The adapter is listed. Select the adapter and click **Start Selected** to start the adapter.

Task B Start the TIBCO ActiveMatrix BusinessWorks Process in TIBCO Administrator

1. In TIBCO Administrator, go to **Application Management**. Click **New Application**. Upload the EAR file created in the previous section and click **OK**. On the following screen, clear the Quick Configure check box and then click **Save**.
2. Click **Configuration** in the newly created application.
3. Expand the `.par` link, so that the TIBCO ActiveMatrix BusinessWorks process is visible.
4. Click on the `.par` and then click the **Add to Additional Machines** button.

5. Select the machine and click **OK**. In the following screen, click **Save**.
6. In the Configuration screen click **Deploy**. In the next screen the Start successfully deployed services check box is selected by default. Click **OK**.
7. Once the deployment is complete, click **Service Instances** under the application. The process is listed. Select the process and click **Start Selected** to start the process.

Task C Trigger and Run the Example

Once the adapter and the TIBCO ActiveMatrix BusinessWorksprocess are started, the following need to be done to trigger the adapter publisher:

1. Open a command window and set the following environment variables before starting the adapter:

- set `TUXDIR=`<Enter the home directory of Tuxedo>
- set `WSNADDR=//`<Enter the Host IP Address with the Port for Workstation Listener>
- set `APPDIR=%TUX_ADAPTER_HOME%\examples\AgentBased\PurchaseOrder`
- `set PATH=%TUXDIR%\bin;%APPDIR%;%PATH%`

2. Go to *TIBCO_HOME*`\adapter\adtuxedo\`*version_number*`\examples\AgentBased\PurchaseOrder`
3. Start the client and specify the `OrderID` argument as follows:

```
Client 1000
```

The client invokes the Tuxedo service. The Tuxedo service sends a request to the adapter through the agent and the agent posts the data to the adapter. The adapter publishes the data over JMS transport. TIBCO ActiveMatrix BusinessWorks subscribes to the data and writes it to the `Get_output.xml`.

Expected Results

The example results can be viewed in the respective XML files written to:

TIBCO_HOME\adapter\adtuxedo*version_number*\examples\AgentBased\PurchaseOrder\BusinessWorks\Get_output.xml

The data published by the adapter is received by TIBCO ActiveMatrix BusinessWorksand written into `Get_output.xml`. A sample output is given below:

```
<?xml version="1.0" encoding="UTF-8"?>
<ns0:PublicationServiceRequest
xmlns:ns0="http://www.tibco.com/xmlns/ae2xsd/2002/05/ae/Tuxedo/TuxedoAdapterConfiguration/PublicationService">
    <POID__33554532>1001</POID__33554532>
    <POTYPE__167772261>Bulk</POTYPE__167772261>
    <UOM__167772262/>
    <PODATE__167772263>29-Jan-2003</PODATE__167772263>
    <CURRENCYTYPE__167772264>Dollars</CURRENCYTYPE__167772264>
    <STATUS__167772265>Active</STATUS__167772265>
    <MFGID__33554538>123</MFGID__33554538>
    <PRICE__33554539>10000</PRICE__33554539>
</ns0:PublicationServiceRequest>
```

Chapter 3

Publication Service for JMS Adapter Agent-Based Communication

This chapter contains a simple example that demonstrates how the adapter's Publication service works using JMS Adapter Agent-based communication paradigm with `FML32` buffers. It also demonstrates the interaction between TIBCO ActiveMatrix BusinessWorks and the adapter in the Publication scenario.

This example uses Java Messaging Service (JMS) as its transport type between the adapter agent and the adapter, and uses TIBCO Rendezvous(RV) as its transport type between the adapter and TIBCO environment.

Topics

- *Example Description, page 22*
- *Setup the Example, page 24*
- *Run and Test the Example, page 25*
- *Expected Results, page 26*

Example Description

This example shows how the adapter Publication service can be used in a TIBCO ActiveMatrix BusinessWorks process using RV transport. The project configured for this example has a TIBCO ActiveMatrix BusinessWorks process which uses adapter Publication service which in turn uses JMS Adapter Agent-based communication.

The business object that is used for this example is `PurchaseOrder`. A Tuxedo service, based on the PurchaseOrder ID that it receives from the invoking client, posts the `POTYPE`, `UOM`, `PODATE`, `CURRENCYTYPE`, `STATUS`, `MFGID`, `PRICE`, `into JMS server`, corresponding to that PurchaseOrder ID. The adapter gets the messages from JMS server using subscriber, and then publishes this data.

Following are the flow of events:

1. User sends request from the Tuxedo client called `Clt`.
2. The client invokes the `POINFO` Tuxedo service.
3. The Tuxedo service puts the data into the JMS server.
4. The adapter gets the data from the JMS server and publishes them over RVCM transport.
5. TIBCO ActiveMatrix BusinessWorks receives the data and writes them in an XML output file.

Location of the Example Files

Files provided with this example are given in the next table. The files are located in the following locations:

TIBCO_HOME`\adapter\adtuxedo\`*version_number*`\examples\AgentBased\JMSTopicPub`

TIBCO_HOME`\adapter\adtuxedo\`*version_number*`\examples\AgentBased\JMSTopicPub`

File Name	Purpose
`Get_input_renderXsd.xsd`	XSD file with XML schema.
`PODetails.h`	Tuxedo header file containing the FML32 definition.

File Name	Purpose
JmsTopicPub.zip	Project zip that contains the adapter Publication service configuration and the TIBCO ActiveMatrix BusinessWorks process. For details on dat files, see Working with dat files in TIBCO Designer on page 4.
adapteragentJMS.h	The header file is needed when compiling the server.c file.
client.c	Source file for the Tuxedo client, which invokes the Tuxedo service, POINFO.
inputdata.txt	File containing input data in XML format.
make.bat	Contains commands required to get the Tuxedo service running on Windows (Sets the required environment variables, creates executables, loads the ubbconfig file and boots the Tuxedo service)
make.sh	Contains commands required to get the Tuxedo service running on UNIX (Sets the required environment variables, creates executables, loads the ubbconfig file and boots the Tuxedo service). This script should be run in a k-shell.
server.c	Source file for Tuxedo Server containing the Tuxedo service, POINFO.
ubbagent	ubbconfig file

Setup the Example

Task A Import the project and save as multi-file project

In TIBCO Designer:

1. In the initial dialog box click **Open New Project** and specify a name for the project.
2. Click **Project>Import Full Project**.
3. Click the Zip Archive tab and browse to locate the PurchaseOrder_Pub_Queue.zip file in the following path,

   TIBCO_HOME\adapter\adtuxedo*version_number*\examples\AgentBased\JMSTopicPub\BusinessWorks
4. Save as a multi-file project.
5. Exit TIBCO Designer.

Task B Start the Tuxedo service

Bring up the sample Tuxedo services given in *TIBCO_HOME*\adapter\adtuxedo*version_number*\examples\AgentBased\JMSTopicPub folder, so that the adapter can communicate with them. See Tasks for Preparing Tuxedo on page 91 for details.

Run and Test the Example

Perform the following tasks to test the example.

Task A Run the Adapter in TIBCO Designer

1. Go to **Tool-> Show Adapter Tester**.
2. In the adapter tester window, click on the instance and in the Run Settings tab give the following parameters:

- `Working Directory`: Specify a directory of your choice
- `Adapter Executable`: Select the adapter executable file

3. Click **Apply** and then **Start**.

Task B Run the TIBCO ActiveMatrix BusinessWorks Process in TIBCO Designer

1. From the project panel, select the process you want to test. In this example the process is `qpub`.
2. Click the **`Set Breakpoints`** icon.
3. In the window that appears, choose **Select All**, then click **OK**.
4. Click the Tester tab to the left of the project panel. The test panel replaces the project tree.
5. Click the **Start testing viewed process** button.
6. In the process selection window that appears, the `qpub` process is selected by default. Click **Load Selected**. The process is now in Test mode. Any change to the input XML file starts the process.
7. Start the client and specify the OrderID argument as follows:

   clt 1000

8. As soon as you start the client, theTIBCO ActiveMatrix BusinessWorks process, `pub` is triggered.
9. Once the process starter is highlighted (indicating a process has started), click the **Step to next activity** icon to step through the process.
10. Click the **Stop Testing** icon to return to design mode.

Expected Results

The example results can be viewed in the respective XML files written to:

TIBCO_HOME\adapter\adtuxedo*version_number*\examples\AgentBased\JMSTopicPub\BusinessWorks

Chapter 4 Publication Service for Tuxedo Queue-Based Communication

This chapter contains a simple example that demonstrates how the adapter's Publication service works using Tuxedo's Queue-based communication paradigm with `FML32` buffers. It also demonstrates the interaction between TIBCO ActiveMatrix BusinessWorksand the adapter in the Publication scenario.

This example uses TIBCO Rendezvous(RV) as its transport type between the adapter and TIBCO environment.

Topics

- *Example Description, page 28*
- *Setup the Example, page 30*
- *Run and Test the Example, page 31*
- *Expected Results, page 32*

Example Description

This example shows how the adapter Publication service can be used in a TIBCO ActiveMatrix BusinessWorksprocess using RV transport. The project configured for this example has a TIBCO ActiveMatrix BusinessWorks process which uses adapter Publication service which in turn uses Queue-based communication.

The business object that is used for this example is `PurchaseOrder`. A Tuxedo service, based on the PurchaseOrder ID that it receives from the invoking client, returns the `POTYPE`, `UOM`, `PODATE`, `CURRENCYTYPE`, `STATUS`, `MFGID`, `PRICE` corresponding to that PurchaseOrder ID. The adapter Publication service publishes this data.

Following are the flow of events:

1. User sends request from the Tuxedo client called `Clt`.
2. The client invokes the `POINFO` Tuxedo service.
3. The Tuxedo service puts the data into the Tuxedo queue.
4. A timer triggers the adapter to check the data in the Tuxedo queue.
5. The adapter gets the data and publishes them over RVCM transport.
6. TIBCO ActiveMatrix BusinessWorks receives the data and writes them in an XML output file.

Location of the Example Files

Files provided with this example are given in the next table. The files are located in the following locations:

TIBCO_HOME`\adapter\adtuxedo\`*version_number*`\examples\QueueBased\PurchaseOrder`

TIBCO_HOME`\adapter\adtuxedo\`*version_number*`\examples\QueueBased\PurchaseOrder\BusinessWorks`

File Name	Purpose
`Get_input_renderXsd.xsd`	XSD file with XML schema.
`PODetails.h`	Tuxedo header file containing the FML32 definition.

File Name	Purpose
PurchaseOrder_Pub_Queue.zip	The project zip that contains the adapter Publication service configuration and the TIBCO ActiveMatrix BusinessWorks process. For details on dat files, see Working with dat files in TIBCO Designer on page 4.
createque.mk	The make file for creating a tuxedo queues and TLOG.
crlog	The file is the shell file to create TLOG.
crque	The file is the shell file to create Tuxedo queues.
rmipc	The file is the shell file to delete queues.
client.c	The source file for the Tuxedo client, which invokes the Tuxedo service, POINFO.
Get_output.xml	The output XML file.
inputdata.txt	File containing input data in XML format.
make.bat	Contains commands required to get the Tuxedo service running on Windows (Sets the required environment variables, creates executables, loads the ubbconfig file and boots the Tuxedo service)
make.sh	Contains commands required to get the Tuxedo service running on UNIX (Sets the required environment variables, creates executables, loads the ubbconfig file and boots the Tuxedo service). This script should be run in a k-shell.
server.c	Source file for Tuxedo Server containing the Tuxedo service, POINFO.
ubbQPub	ubbconfig file

Setup the Example

Task A Import the project and save as multi-file project

In TIBCO Designer:

1. In the initial dialog box click **Open New Project** and specify a name for the project.
2. Click **Project>Import Full Project**.
3. Click the Zip Archive tab and browse to locate the PurchaseOrder_Pub_Queue.zip file in the following path,

   TIBCO_HOME\adapter\adtuxedo*version_number*\examples\QueueBased\PurchaseOrder\BusinessWorks
4. Save as a multi-file project.
5. Exit TIBCO Designer.

Task B Start the Tuxedo service

Bring up the sample Tuxedo services given in *TIBCO_HOME*\adapter\adtuxedo*version_number*\examples\QueueBased\PurchaseOrder folder, so that the adapter can communicate with them. See Tasks for Preparing Tuxedo on page 91 for details. In this example, after the `make.bat` file is run, the Tuxedo queues are created.

Run and Test the Example

Perform the following tasks to test the example.

Task A Run the Adapter in TIBCO Designer

1. Go to **Tool-> Show Adapter Tester**.
2. In the adapter tester window, click on the instance and in the Run Settings tab give the following parameters:

- `Working Directory`: Specify a directory of your choice
- `Adapter Executable`: Select the adapter executable file

3. Click **Apply** and then **Start**.

Task B Run the TIBCO ActiveMatrix BusinessWorks Process in TIBCO Designer

1. From the project panel, select the process you want to test. In this example the process is `qpub`.
2. Click the **`Set Breakpoints`** icon.
3. In the window that appears, choose **Select All**, then click **OK**.
4. Click the Tester tab to the left of the project panel. The test panel replaces the project tree.
5. Click the **Start testing viewed process** button.
6. In the process selection window that appears, the `qpub` process is selected by default. Click **Load Selected**. The process is now in Test mode. Any change to the input XML file starts the process.
7. Start the client and specify the OrderID argument as follows:

   clt 1000

8. As soon as you start the client, the TIBCO ActiveMatrix BusinessWorks process, `qpub` is triggered.
9. Once the process starter is highlighted (indicating a process has started), click the **Step to next activity** icon to step through the process.
10. Click the **Stop Testing** icon to return to design mode.

Expected Results

The example results can be viewed in the respective XML files written to:

TIBCO_HOME\adapter\adtuxedo*version_number*\examples\QueueBased\PurchaseOrder\BusinessWorks\Get_output.xml

The data published by the adapter is received by TIBCO ActiveMatrix BusinessWorks and written into `Get_output.xml`. A sample output is given below:

```
<?xml version="1.0" encoding="UTF-8"?>
<ns0:PublicationServiceRequest
xmlns:ns0="http://www.tibco.com/xmlns/ae2xsd/2002/05/ae/Tuxedo/TuxedoAdapterConfiguration/PublicationService">
    <POID__33554532>1001</POID__33554532>
    <POTYPE__167772261>Bulk</POTYPE__167772261>
    <UOM__167772262/>
    <PODATE__167772263>29-Jan-2003</PODATE__167772263>
    <CURRENCYTYPE__167772264>Dollars</CURRENCYTYPE__167772264>
    <STATUS__167772265>Active</STATUS__167772265>
    <MFGID__33554538>123</MFGID__33554538>
    <PRICE__33554539>10000</PRICE__33554539>

</ns0:PublicationServiceRequest>
```

Chapter 5 Subscription Service for Asynchronous Communication

This chapter contains a simple example that demonstrates how the adapter's Subscription service works using Tuxedo's asynchronous communication paradigm with `FML32` buffers. It also demonstrates the interaction between TIBCO ActiveMatrix BusinessWorks and the adapter in the Subscription scenario.

This example uses Java Messaging Service (JMS) as its transport type between the adapter and TIBCO environment.

Topics

- *Example Description, page 34*
- *Setup the Example, page 37*
- *Run and Test the Example, page 38*
- *Expected Results, page 40*

Example Description

This example shows how the adapter Subscription service can be used in a TIBCO ActiveMatrix BusinessWorks process using JMS transport. The project configured for this example has a TIBCO ActiveMatrix BusinessWorks process which sends data on the subject that the adapter Subscription service is subscribing to. The adapter Subscription service, in turn uses the Asynchronous communication.

The example simulates the Asynchronous communication paradigm that is supported by the adapter Subscription service. The Tuxedo service, based on the CUSTID, NAME, ADDRESS, CITY, STATE, ZIP and PHONE data that it receives from the invoking adapter, writes the data into a flat file named Customer. It also generates an ID for that customer and includes it in a file named GeneratedCustomerIds.

Following are the flow of events:

1. User enters values for the following input data into the Get_input.xml.

   CUSTID, NAME, ADDRESS, CITY, STATE, ZIP, PHONE, ServiceName, CommunicationType in the Get_input.xml as shown below:

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

<Customers
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <Customer>

    <CUSTID>15</CUSTID>

    <NAME>Arundhati</NAME>

    <ADDRESS>Vijaya Enclave</ADDRESS>

    <CITY>Bangalore</CITY>

    <STATE>Karnataka</STATE>

    <ZIP>560098</ZIP>

    <PHONE>1234567</PHONE>

    <ServiceName>NEWCUST</ServiceName>

    <CommunicationType>Async</CommunicationType>

  </Customer>

</Customers>
```

2. The TIBCO ActiveMatrix BusinessWorks process, sub gets triggered. The file poller in the process reads the data and forwards it to the publisher which publishes the data.
3. The Subscription service, which is listening on the same subject as that of the invoking client (TIBCO ActiveMatrix BusinessWorks), subscribes to the subject and gets the data.
4. The Subscription service in turn invokes the Tuxedo service NEWCUST.
5. The Tuxedo services NEWCUST, based on the data that it receives from the Subscription service, writes the data to CustomerDetails file.

Location of the Example Files

Files provided with this example are given in the next table. The files are located in the following locations:

TIBCO_HOME\adapter\adtuxedo*version_number*\examples\Asynchronous\Customer

TIBCO_HOME\adapter\adtuxedo*version_number*\examples\Asynchronous\Customer\BusinessWorks

File Name	Purpose
CustDetails.h	Tuxedo header file containing the FML32 definition.
Customer_Sub_Async.zip	Project zip that contains the preconfigured adapter Subscription service and the TIBCO ActiveMatrix BusinessWorks process. For details on dat files, see Working with dat files in TIBCO Designer on page 4.
Get_input.xml	File containing input data in XML format.
Get_input_parserXsd.xsd	XSD to validate the input XML.
Get_output.xml	The output XML file.
Services.h	Header file required by the sample Tuxedo server program.
make.bat	Contains commands required to get the Tuxedo service running on Windows (Sets the required environment variables, creates executables, loads the ubbconfig file and boots the Tuxedo service).

File Name	Purpose
`make.sh`	Contains commands required to get the Tuxedo service running on UNIX (Sets the required environment variables, creates executables, loads the `ubbconfig` file and boots the Tuxedo service). This script should be run in a k-shell.
`server.c`	Source file for Tuxedo Server containing the service `NEWCUSTID`
`ubbAsyn`	`ubbconfig` file
`util.c`	Source file used by the Tuxedo service

Setup the Example

Task A Import the project and save as multi-file project

In TIBCO Designer:

1. In the initial dialog box click **Open New Project** and specify a name for the project.
2. Click **Project>Import Full Project**.
3. Click the Zip Archive tab and browse to locate the `Customer_Sub_Async.zip` file in the following path,

   TIBCO_HOME`\adapter\adtuxedo\`*version_number*`\examples\Asynchronous\Customer\BusinessWorks`
4. Save as a multi-file project.
5. Exit TIBCO Designer.

Task B Start EMS Server

This example uses JMS as the transport type. Ensure that the TIBCO Enterprise Message Service server is running and accessible to the machine on which the adapter is installed before running any of the processes.

Run the `tibemsd.exe` file from your *TIBCO_HOME*`/EMS/bin` directory

Task C Start the Tuxedo service

Bring up the sample Tuxedo services given in *TIBCO_HOME*`\adapter\adtuxedo\`*version_number*`\examples\Asynchronous\Customer` folder, so that the adapter can communicate with them. See Tasks for Preparing Tuxedo on page 91 for details.

Run and Test the Example

Perform the following tasks to test the example.

Task A Run the Adapter in TIBCO Designer

1. Go to **Tool-> Show Adapter Tester**.
2. In the adapter tester window, click on the instance and in the Run Settings tab give the following parameters:

- `Working Directory`: Specify a directory of your choice
- `Adapter Executable`: Select the adapter executable file

3. Click **Apply** and then **Start**.

Task B Run the TIBCO ActiveMatrix BusinessWorks Process in TIBCO Designer

1. From the project panel, select the process you want to test. In this example the process is `sub`.
2. Click the **`Set Breakpoints`** icon.
3. In the window that appears, choose **Select All**, then click **OK**.
4. Click the Tester tab to the left of the project panel. The test panel replaces the project tree.
5. Click the **Start testing viewed process** button.
6. In the process selection window that appears, the `sub` process is selected by default. Click **Start Selected**. The process is now in Test mode. Any change to the input XML file starts the process.
7. Enter or change the values for `CUSTID`, `NAME`, `ADDRESS`, `CITY`, `STATE`, `ZIP`, `PHONE`, `ServiceName`, `CommunicationType` in the `Get_input.xml` as shown below:

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

<Customers
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <Customer>

    <CUSTID>15</CUSTID>

    <NAME>Arundhati</NAME>
```

```
    <ADDRESS>Vijaya Enclave</ADDRESS>
    <CITY>Bangalore</CITY>
    <STATE>Karnataka</STATE>
    <ZIP>560098</ZIP>
    <PHONE>1234567</PHONE>
    <ServiceName>NEWCUST</ServiceName>
    <CommunicationType>Async</CommunicationType>
  </Customer>
</Customers>
```

8. As soon as you make any change to the `Get_input.xml` process the TIBCO ActiveMatrix BusinessWorks process, `Sub` is triggered.
9. Once the process starter is highlighted (indicating a process has started), click the **Step to next activity** icon to step through the process.
10. Click the **Stop Testing** icon to return to design mode.

Expected Results

The Subscription service subscribes to the published data and invokes the NEWCUST Tuxedo service.

The NEWCUST service takes the details and writes them to the CustomerDetails file in the following format:

```
15: Arundhati: Vijaya Enclave: Bangalore: Karnataka: 560098:
1234567
```

Chapter 6 Subscription Service for Conversational Communication

This chapter contains a simple example that demonstrates how the adapter's Subscription service works using Tuxedo's Conversational communication paradigm with `FML32` buffers. It also demonstrates the interaction between TIBCO ActiveMatrix BusinessWorks and the adapter in the Subscription scenario.

This example uses TIBCO Rendezvous (RV) as its transport type between the adapter and TIBCO environment.

Topics

- *Example Description, page 42*
- *Setup the Example, page 45*
- *Run and Test the Example, page 46*
- *Expected Results, page 48*

Example Description

This example shows how the adapter Subscription service can be used in a TIBCO ActiveMatrix BusinessWorks process using TIBCO Rendezvous transport. The project configured for this example has a TIBCO ActiveMatrix BusinessWorks process, which sends data on the subject that the adapter Subscription service is subscribing to. The adapter Subscription service uses the Conversational communication.

The example simulates the Conversational communication paradigm that is supported by the adapter Subscription service. The Tuxedo service, based on the value for `OCCURS` (Batch Size) that it receives from the invoking client, returns the records (that will contain `ITEMCODE`, `ITEMID`, `PRICE`) in batches (If `OCCURS = 2`, it will return 2 records at a time).

Following are the flow of events:

1. User enters the input data (values for `OCCURS`) into the `Get_input.xml` as follows:

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

<Materials
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <Material>

    <OCCURS>1</OCCURS>

  </Material>

</Materials>
```

2. The TIBCO ActiveMatrix BusinessWorks process, `sub-send` gets triggered. The file poller in the process reads the data and forwards it to the publisher which publishes the data.
3. The Subscription service, which is listening on the same subject as that of the invoking client (TIBCO ActiveMatrix BusinessWorks), subscribes to the subject and gets the data.
4. The Subscription service in turn invokes the Tuxedo service, `SEARCHITEM`.
5. The service based on the `BatchSize` that is received retrieves the records in batches from the `items` file.
6. The Subscription service receives these records and sends these on the reply subject.
7. The `sub-reply` process in TIBCO ActiveMatrix BusinessWorks gets triggered once the reply is received.

8. The data received on the reply subject is then written to an XML namely `Get_output.xml`.

Location of the Example Files

Files provided with this example are given in the next table. The files are located in the following locations:

TIBCO_HOME\adapter\adtuxedo*version_number*\examples\Conversational\Material

TIBCO_HOME\adapter\adtuxedo*version_number*\examples\Conversational\Material\BusinessWorks

File Name	Purpose
`Get_input.xml`	File containing input data in XML format.
`Get_input_parserXsd.xsd`	XSD containing input schema definition.
`Get_input_renderXsd.xsd`	XSD containing output schema definition.
`Get_output.xml`	The output XML file.
`MaterialInfoInput.h`	Tuxedo header file containing the FML32 definition for the Subscription service.
`MaterialInfoOutput.h`	Tuxedo header file containing the FML32 definition for the Subscription reply.
`Material_Sub_Conv.zip`	The project zip that contains the preconfigured adapter subscription service and theTIBCO ActiveMatrix BusinessWorks process. For details on dat files, see Working with dat files in TIBCO Designer on page 4.
`items`	This file (instead of a database) contains the data that will be used by the Tuxedo service.
`make.bat`	Contains commands required to get the Tuxedo service running on Windows (Sets the required environment variables, creates executables, loads the `ubbconfig` file and boots the Tuxedo service).

File Name	Purpose
`make.sh`	Contains commands required to get the Tuxedo service running on UNIX (Sets the required environment variables, creates executables, loads the `ubbconfig` file and boots the Tuxedo service). This script should be run in a k-shell.
`server.c`	Source file for Tuxedo server containing the service `SEARCHITEM`
`ubbconv`	`ubbconfig` file
`util.c`	Source file used by the Tuxedo service.

Setup the Example

Task A Import the project and save as multi-file project

In TIBCO Designer:

1. In the initial dialog box click **Open New Project** and specify a name for the project.
2. Click **Project>Import Full Project**.
3. Click the Zip Archive tab and browse to locate the `Material_Sub_Conv.zip` file in the following path,

   TIBCO_HOME`\adapter\adtuxedo\`*version_number*`\examples\Conversational\Material\BusinessWorks`
4. Save as a multi-file project.
5. Exit TIBCO Designer.

Task B Start the Tuxedo service

Bring up the sample Tuxedo services given in *TIBCO_HOME*`\adapter\adtuxedo\`*version_number*`\examples\Conversational\Material` folder, so that the adapter can communicate with them. See Tasks for Preparing Tuxedo on page 91 for details.

Run and Test the Example

Perform the following tasks to test the example.

Task A Run the Adapter in TIBCO Designer

1. Go to **Tool-> Show Adapter Tester**.
2. In the adapter tester window, click on the instance and in the Run Settings tab give the following parameters:
   - `Working Directory`: Specify a directory of your choice
   - `Adapter Executable`: Select the adapter .exe file (`adtuxedo_native.exe` or `adtuxedo_wrkstn.exe`)
3. Click **Apply** and then **Start**.

Task B Run the TIBCO ActiveMatrix BusinessWorksProcess in TIBCO Designer

1. From the project panel, select the process you want to test. For example, `sub-send`.
2. Click the **`Set Breakpoints`** icon.
3. In the window that appears, choose **Select All**, then click **OK**.
4. Click the Tester tab to the left of the project panel. The test panel replaces the project tree.
5. Click the **Start testing viewed process** button.
6. In the process selection window that appears, the `sub-send` process is selected by default. Click **Start Selected**. The process is now in Test mode. Any change to the input XML file starts the process.
7. Enter or change the values for `OCCURS` in the `Get_input.xml` in the format shown below:

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

<Materials
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <Material>

    <OCCURS>1</OCCURS>

  </Material>
```

8. As soon as you make changes to the `Get_input.xml` file, the TIBCO ActiveMatrix BusinessWorks process, `sub-send` is triggered.
9. Once the process starter is highlighted (indicating a process has started), click the **Step to next activity** icon to step through the process.
10. The `sub-reply` TIBCO ActiveMatrix BusinessWorks process gets triggered once the reply is received from the adapter Subscription service. The `sub-reply` process writes data into the `output.xml` file.
11. Click the **Stop Testing** icon to return to design mode.

Expected Results

Results for this example will be written into `Get_output.xml` file located at `D:\examples\Conversational\Material\BusinessWorks` folder. A sample output is given next.

```
<?xml version="1.0" encoding="UTF-8"?>
<Material>
    <SubscriptionServiceReplySchema>
        <OCCURS>6</OCCURS>
        <ITEMCODE>1000</ITEMCODE>
        <ITEMID>tablet 0</ITEMID>
        <PRICE>20.000000</PRICE>
    </SubscriptionServiceReplySchema>
</Material><?xml version="1.0" encoding="UTF-8"?>
<Material>
    <SubscriptionServiceReplySchema>
        <OCCURS>6</OCCURS>
        <ITEMCODE>1001</ITEMCODE>
        <ITEMID>tablet 1</ITEMID>
        <PRICE>21.000000</PRICE>
    </SubscriptionServiceReplySchema>
</Material><?xml version="1.0" encoding="UTF-8"?>
<Material>
    <SubscriptionServiceReplySchema>
        <OCCURS>6</OCCURS>
        <ITEMCODE>1002</ITEMCODE>
        <ITEMID>tablet 2</ITEMID>
        <PRICE>22.000000</PRICE>
    </SubscriptionServiceReplySchema>
</Material><?xml version="1.0" encoding="UTF-8"?>
<Material>
    <SubscriptionServiceReplySchema>
        <OCCURS>6</OCCURS>
        <ITEMCODE>1003</ITEMCODE>
        <ITEMID>tablet 3</ITEMID>
        <PRICE>23.000000</PRICE>
    </SubscriptionServiceReplySchema>
</Material><?xml version="1.0" encoding="UTF-8"?>
<Material>
    <SubscriptionServiceReplySchema>
        <OCCURS>6</OCCURS>
        <ITEMCODE>1004</ITEMCODE>
        <ITEMID>tablet 4</ITEMID>
        <PRICE>24.000000</PRICE>
    </SubscriptionServiceReplySchema>
</Material><?xml version="1.0" encoding="UTF-8"?>
<Material>
    <SubscriptionServiceReplySchema>
        <OCCURS>6</OCCURS>
        <ITEMCODE>1005</ITEMCODE>
        <ITEMID>tablet 5</ITEMID>
        <PRICE>25.000000</PRICE>
    </SubscriptionServiceReplySchema>
```

```
</Material><?xml version="1.0" encoding="UTF-8"?>
<Material>
    <SubscriptionServiceReplySchema>
        <OCCURS>4</OCCURS>
        <ITEMCODE>1006</ITEMCODE>
        <ITEMID>tablet 6</ITEMID>
        <PRICE>26.000000</PRICE>
    </SubscriptionServiceReplySchema>
</Material><?xml version="1.0" encoding="UTF-8"?>
<Material>
    <SubscriptionServiceReplySchema>
        <OCCURS>4</OCCURS>
        <ITEMCODE>1007</ITEMCODE>
        <ITEMID>tablet 0</ITEMID>
        <PRICE>20.000000</PRICE>
    </SubscriptionServiceReplySchema>
</Material><?xml version="1.0" encoding="UTF-8"?>
<Material>
    <SubscriptionServiceReplySchema>
        <OCCURS>4</OCCURS>
        <ITEMCODE>1008</ITEMCODE>
        <ITEMID>tablet 1</ITEMID>
        <PRICE>21.000000</PRICE>
    </SubscriptionServiceReplySchema>
</Material><?xml version="1.0" encoding="UTF-8"?>
<Material>
    <SubscriptionServiceReplySchema>
        <OCCURS>4</OCCURS>
        <ITEMCODE>1009</ITEMCODE>
        <ITEMID>tablet 2</ITEMID>
        <PRICE>22.000000</PRICE>
    </SubscriptionServiceReplySchema>
</Material><?xml version="1.0" encoding="UTF-8"?>
```

Chapter 7 Subscription Service for Queue-Based Communication

This chapter contains a simple example that demonstrates how the adapter's Subscription service works using Tuxedo's Queue-based communication paradigm with `FML32` buffers. It also demonstrates the interaction between TIBCO ActiveMatrix BusinessWorks and the adapter in the Subscription scenario.

This example uses Java Messaging Service (JMS) as its transport type between the adapter and TIBCO environment.

Topics

- *Example Description, page 52*
- *Setup the Example, page 55*
- *Run and Test the Example, page 56*
- *Expected Results, page 58*

Example Description

This example shows how the adapter Subscription service can be used in a TIBCO ActiveMatrix BusinessWorks process using JMS transport. The project configured for this example has a TIBCO ActiveMatrix BusinessWorks process which sends data on the subject that the adapter Subscription service is subscribing to. The adapter Subscription service uses the Queue-based communication.

The example simulates the Queue-based communication paradigm that is supported by the adapter Subscription service. The Tuxedo service, based on the `Department ID` that it receives from the invoking client, returns the `Employee ID`, `Employee Name` and `Department Name` corresponding to that `Department ID`.

Following are the flow of events:

1. Enter the values for `Department ID`, `ReplyQueue`, `CommunicationType`, `ServiceName`, `QueueSpace` in the `Get_input.xml` as shown below:

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

<Employees
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <Employee>

    <DEPTID>15</DEPTID>

    <ReplyQueue>REPLYQ</ReplyQueue>

    <CommunicationType>Queue</CommunicationType>

    <ServiceName>QSERVICENEW</ServiceName>

    <QueueSpace>QSPACE</QueueSpace>

  </Employee>

</Employees>
```

2. The TIBCO ActiveMatrix BusinessWorks process, `sub-send` gets triggered. The file poller in the process reads the data and forwards it to the publisher which publishes the data.
3. The Subscription service, which is listening on the same subject as that of the invoking client (TIBCO ActiveMatrix BusinessWorks, subscribes to the subject and gets the data.
4. The Subscription service in turn invokes the Tuxedo Service, `EMPINFO`.
5. The service based on the `Department ID` that is received, queries the flat file - `inputdata.txt` for all employee records with matching `Department ID`.

6. The Subscription service sends the records that matched the query on the reply subject.
7. The `sub-receive` process in TIBCO ActiveMatrix BusinessWorks gets triggered once the reply is received.
8. The data received on the reply subject is then written to the `Get_output.xml` file.

Location of the Example Files

Files provided with this example are given in the next table. The files are located in the following locations:

TIBCO_HOME`\adapter\adtuxedo\`*version_number*`\examples\QueueBased\Employee`

TIBCO_HOME`\adapter\adtuxedo\`*version_number*`\examples\QueueBased\Employee\BusinessWorks`

File Name	Purpose
`Create Queues.txt`	Specifies the commands to create the Queues that need to be created.
`EmployeeDetails.h`	Tuxedo header file containing the FML32 definition for the Subscription service.
`Employee_Sub_Queue.zip`	Project zip that contains the preconfigured adapter Subscription service and the TIBCO ActiveMatrix BusinessWorks process. For details on dat files, see Working with dat files in TIBCO Designer on page 4.
`Get_input.xml`	File containing input data in XML format.
`Get_input_parserXsd.xsd`	XSD containing input schema definition.
`Get_input_renderXsd.xsd`	XSD containing output schema definition.
`inputdata.txt`	This file (instead of a database) contains the data that will be used by the server service.
`make.bat`	Contains commands required to get the Tuxedo service running on Windows (Sets the required environment variables, creates executables, loads the `ubbconfig` file and boots the Tuxedo service)

File Name	Purpose
`make.sh`	Contains commands required to get the Tuxedo service running on UNIX (Sets the required environment variables, creates executables, loads the `ubbconfig` file and boots the Tuxedo service). This script should be run in a k-shell.
`server.c`	Source file for Tuxedo Server containing the service, `QSERVICENEW`
`ubbQ`	`ubbconfig` file

Setup the Example

Task A Import the project and save as multi-file project

In TIBCO Designer:

1. In the initial dialog box click **Open New Project** and specify a name for the project.
2. Click **Project>Import Full Project**.
3. Click the Zip Archive tab and browse to locate the `Employee_Sub_Queue.zip` file in the following path,

   TIBCO_HOME`\adapter\adtuxedo\`*version_number*`\examples\QueueBased\Employee\BusinessWorks`
4. Save as a multi-file project.
5. Exit TIBCO Designer.

Task B Start EMS Server

This example uses JMS as the transport type. Ensure that the TIBCO Enterprise Message Service server is running and accessible to the machine on which the adapter is installed before running any of the processes.

Run the `tibemsd.exe` file from your *TIBCO_HOME*`/EMS/bin` directory.

Task C Start the Tuxedo service

Bring up the sample Tuxedo services given in *TIBCO_HOME*`\adapter\adtuxedo\`*version_number*`\examples\QueueBased\Employee` folder, so that the adapter can communicate with them. See Tasks for Preparing Tuxedo on page 91 for details.

Run and Test the Example

Perform the following tasks to test the example.

Task A Run the Adapter in TIBCO Designer

1. Go to **Tool-> Show Adapter Tester**.
2. In the adapter tester window, click on the instance and in the Run Settings tab give the following parameters:
   - `Working Directory`: Specify a directory of your choice
   - `Adapter Executable`: Select the adapter .exe file (`adtuxedo_native.exe` or `adtuxedo_wrkstn.exe`)
3. Click **Apply** and then **Start**.

Task B Run the TIBCO ActiveMatrix BusinessWorks Process in TIBCO Designer

1. From the project panel, select the process you want to test. For example, `sub-send`.
2. Click the **`Set Breakpoints`** icon.
3. In the window that appears, choose **Select All**, then click **OK**.
4. Click the Tester tab to the left of the project panel. The test panel replaces the project tree.
5. Click the **Start testing viewed process** button.
6. In the process selection window that appears, the `sub-send` process is selected by default. Click **Start Selected**. The process is now in Test mode. Any change to the input XML file starts the process.
7. Enter the values for `Department ID`, `ReplyQueue`, `CommunicationType`, `ServiceName`, `QueueSpace` in the `Get_input.xml` as shown below:

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

<Employees
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <Employee>

    <DEPTID>15</DEPTID>

    <ReplyQueue>REPLYQ</ReplyQueue>
```

```
    <CommunicationType>Queue</CommunicationType>
    <ServiceName>QSERVICENEW</ServiceName>
    <QueueSpace>QSPACE</QueueSpace>
  </Employee>
</Employees>
```

8. As soon as you make a change in the `Get_input.xml` the TIBCO ActiveMatrix BusinessWorks process, `sub-send` is triggered.
9. Once the process starter is highlighted (indicating a process has started), click the **Step to next activity** icon to step through the process.
10. The `sub-receive`TIBCO ActiveMatrix BusinessWorkss process gets triggered once the reply is received from the adapter Subscription service. The `sub-receive` process writes data into the `output.xml` file.
11. Click the **Stop Testing** icon to return to design mode.

Expected Results

Results for this example will be written into `Get_output.xml` file located at `D:\examples\QueueBased\Employee\BusinessWorks` folder. A sample output is given next.

```
<?xml version="1.0" encoding="UTF-8"?>
<Employee>
    <SubscriptionServiceReplySchema>
        <SubscriptionServiceReply_Account_List>
            <EMPID>4856</EMPID>
            <EMPNAME>12</EMPNAME>
            <DEPTID>balaji sc athreye</DEPTID>
            <DEPTNAME>enterprise solutions
</DEPTNAME>
        </SubscriptionServiceReply_Account_List>
    </SubscriptionServiceReplySchema>
</Employee>
```

Chapter 8 Request-Response Service for Synchronous Communication

This chapter contains a simple example that demonstrates how the adapter's Request-Response service works using Tuxedo's Synchronous communication paradigm with `FML32` buffers. It also demonstrates the interaction between TIBCO ActiveMatrix BusinessWorks and the adapter in the Request-Response scenario.

This example uses TIBCO Rendezvous (RV) as its transport type between the adapter and TIBCO environment.

Topics

- *Example Description, page 60*
- *Setup the Example, page 63*
- *Run and Test the Example, page 64*
- *Expected Results, page 66*

Example Description

This example simulates the Synchronous communication paradigm that is supported by the adapter Request-Response service. The Tuxedo service, based on the DEPTID that it receives from the invoking client (the Request-Response service in this case), returns the EMPID, EMPNAME and DEPTID corresponding to that DEPTID.

This example can be run in two ways:

1. Loading Field ID Dynamically at run-time — The Field ID for FML32 fields are loaded dynamically from the corresponding field tables at runtime.
2. Loading the Field ID stored in the repository — The Field IDs for FML32 fields are read from the corresponding header files and stored in the repository.

Following are the flow of events:

1. Enter the values for Department ID in the Get_input.xml as shown below:

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

<Employees
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <Employee>

    <DEPTID>123</DEPTID>

  </Employee>

</Employees>.
```

2. TIBCO ActiveMatrix BusinessWorks reads the input data and sends the request. The TIBCO ActiveMatrix BusinessWorks process, rpc gets triggered. The file poller in the process reads the data and forwards it to the Request-Response Invocation service in the process which publishes the data.
3. The Request-Response service, which is listening on the same subject as that of the invoking client (TIBCO ActiveMatrix BusinessWorks), receives the request.
4. The Request-Response service then invokes the Tuxedo Service EMPINFO.
5. The Tuxedo service based on the Department ID that is received, queries the flat file - inputdata.txt.
6. The Tuxedo service then returns the records that match the query.
7. The TIBCO ActiveMatrix BusinessWorks process, rpc then writes this data into the Get_output.xml.

Location of the Example Files

Files provided with this example are given in the next table. The files are located in the following locations:

For the Static example:

TIBCO_HOME\adapter\adtuxedo*version_number*\examples\Synchronous\Static\Employee\BusinessWorks

For the Dynamic example:

TIBCO_HOME\adapter\adtuxedo*version_number*\examples\Synchronous\Dynamic\Employee\BusinessWorks

File Name	Purpose
EmployeeDetailsInput.h	Tuxedo header file containing the FML32 definition for the request buffer.
EmployeeDetailsOutput.h	Tuxedo header file containing the FML32 definition for the response buffer.
Employee_RPC_Sync.zip	Project zip that contains the adapter request-response configuration and the TIBCO ActiveMatrix BusinessWorks process. For details on dat files, see Working with dat files in TIBCO Designer on page 4.
Get_input.xml	File containing request data in XML format.
Get_input_parserXsd.xsd	XSD containing input schema definition.
Get_input_renderXsd.xsd	XSD containing output schema definition.
Get_output.xml	Output XML file.
inputdata.txt	This file (instead of a database) contains the data that will be used by the server service.
make.bat	Contains commands required to get the Tuxedo service running on Windows (Sets the required environment variables, creates executables, loads the ubbconfig file and boots the Tuxedo service)

File Name	Purpose
`make.sh`	Contains commands required to get the Tuxedo service running on UNIX (Sets the required environment variables, creates executables, loads the `ubbconfig` file and boots the Tuxedo service). This script should be run in a k-shell.
`server.c`	Source file for Tuxedo Server containing the service `EMPINFO`
`ubbrpc`	`ubbconfig` file
`EmployeeDetailsInput.fml`	Tuxedo field table file containing the FML32 definition for the request buffer.
`EmployeeDetailsOutput.fml`	Tuxedo field table file containing the FML32 definition for the reply buffer.

Setup the Example

Task A Import the project and save as multi-file project

In TIBCO Designer:

1. In the initial dialog box click **Open New Project** and specify a name for the project.
2. Click **Project>Import Full Project**.
3. Click the Zip Archive tab and browse to locate the `Employee_RPC_Sync.zip` file in the following paths,

   For the Static example:

   TIBCO_HOME`\adapter\adtuxedo\`*version_number*`\examples\Synchronous\Static\Employee\BusinessWorks`

   For the Dynamic example:

   TIBCO_HOME\adapter\adtuxedo*version_number*\examples\Synchronous\Dynamic\Employee\BusinessWorks
4. Save as a multi-file project.
5. Exit TIBCO Designer.

Task B Start the Tuxedo service

Bring up the sample Tuxedo services given in the following folders as required so that the adapter can communicate with them:

For the Static example:

TIBCO_HOME`\adapter\adtuxedo\`*version_number*`\examples\Synchronous\Static\Employee\BusinessWorks`

For the Dynamic example:

TIBCO_HOME`\adapter\adtuxedo\`*version_number*`\examples\Synchronous\Dynamic\Employee\BusinessWorks`

See Tasks for Preparing Tuxedo on page 91 for details.

Run and Test the Example

Perform the following tasks to test the example.

Task A Run the Adapter in TIBCO Designer

1. Go to **Tool-> Show Adapter Tester**.
2. In the adapter tester window, click on the instance and in the Run Settings tab give the following parameters:

- `Working Directory`: Specify a directory of your choice
- `Adapter Executable`: Select the adapter .exe file (`adtuxedo_native.exe` or `adtuxedo_wrkstn.exe`)

3. Click **Apply** and then **Start**.

Task B Run the TIBCO ActiveMatrix BusinessWorks Process in TIBCO Designer

1. From the project panel, select the process you want to test. In this example the process is `rpc`.
2. Click the **`Set Breakpoints`** icon.
3. In the window that appears, choose **Select All**, then click **OK**.
4. Click the Tester tab to the left of the project panel. The test panel replaces the project tree.
5. Click the **Start testing viewed process** button.
6. In the process selection window that appears, the `rpc` process is selected by default. Click **Start Selected**. The process is now in Test mode. Any change to the input XML file starts the process.
7. Enter or change the values for `Department ID` in the `Get_input.xml` as shown below:

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

<Employees
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <Employee>

    <DEPTID>123</DEPTID>

  </Employee>
```

```
</Employees>
```

8. As soon as you make a change in the `Get_input.xml` the TIBCO ActiveMatrix BusinessWorks process, `rpc` is triggered.
9. Once the process starter is highlighted (indicating a process has started), click the **Step to next activity** icon to step through the process.
10. The TIBCO ActiveMatrix BusinessWorks process, `rpc` then writes the data received from the Tuxedo service into the `Get_output.xml`.
11. Click the **Stop Testing** icon to return to design mode.

Expected Results

Results for this example will be written into `Get_output.xml` file located in the following locations:

For the Static example:

TIBCO_HOME`\adapter\adtuxedo\`*version_number*`\examples\Synchronous\Static\Employee\BusinessWorks`

For the Dynamic example:

TIBCO_HOME`\adapter\adtuxedo\`*version_number*`\examples\Synchronous\Dynamic\Employee\BusinessWorks`

A sample output is given next.

```
<?xml version="1.0" encoding=""?>
<ns0:xmlInput
xmlns:ns0="http://xmlns.tibco.com/bw/activity/xml/render/bytesEnve
lope/2003/05">
    <Employee>
        <RequestResponseReplySchema>
            <RequestResponseServiceReply_a_List>
                <EMPID>4856</EMPID>
                <EMPNAME>John Smith</EMPNAME>
                <DEPTID>123</DEPTID>
                <DEPTNAME>IT Solutions</DEPTNAME>
            </RequestResponseServiceReply_a_List>
        </RequestResponseReplySchema>
    </Employee>
</ns0:xmlInput>
```

Chapter 9 Request-Response Service for Asynchronous Communication

This chapter contains a simple example that demonstrates how the adapter's Request-Response service works using Tuxedo's Asynchronous communication paradigm with `FML32` buffers. It also demonstrates the interaction between TIBCO ActiveMatrix BusinessWorks and the adapter in the Request-Response scenario.

This example uses Java Messaging Service (JMS) as its transport type between the adapter and TIBCO environment.

Topics

- *Example Description, page 68*
- *Setup the Example, page 70*
- *Run and Test the Example, page 71*
- *Expected Results, page 73*

Example Description

This example simulates the Asynchronous communication paradigm that is supported by the adapter Request-Response Service. The Tuxedo service, based on the PRODUCTID that it received from the data sent by the invoking client, returns the CATEGORYID, PRODUCTNAME, DESCRIPTION, UOM, PRICE corresponding to that PRODUCTID.

Following are the flow of events:

1. Enter the values for Product ID and ServiceName in the Get_input.xml as shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<Products xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Product>
    <PRODUCTID>1</PRODUCTID>
    <ServiceName>PRODINFO</ServiceName>
  </Product>
</Products>.
```

2. The TIBCO ActiveMatrix BusinessWorks process, rpc reads the input data and sends the request.
3. The Request-Response service, which is listening on the same subject as that of the invoking client (TIBCO ActiveMatrix BusinessWorks), receives the request.
4. The Request-Response service invokes the Tuxedo Service PRODINFO.
5. The Tuxedo service, based on the PRODUCTID that is received, queries the flat file - inputdata.txt.
6. The Tuxedo service then returns the records that match the query.
7. The TIBCO ActiveMatrix BusinessWorks process, rpc then writes this data into the Get_output.xml.

Location of the Example Files

Files provided with this example are given in the next table. The files are located in the following locations:

TIBCO_HOME\adapter\adtuxedo*version_number*\examples\Asynchronous\Product

TIBCO_HOME\adapter\adtuxedo*version_number*\examples\Asynchronous\Product\BusinessWorks

File Name	Purpose
Get_input.xml	File containing request data in XML format.
Get_input_parserXsd.xsd	XSD containing input schema definition.
Get_input_renderXsd.xsd	XSD containing output schema definition.
Get_output.xml	The output XML file.
ProductInput.h	Tuxedo header file containing the FML32 definition for the request buffer.
ProductOutput.h	Tuxedo header file containing the FML32 definition for the response buffer.
Product_RPC_Async.zip	Project zip that contains the preconfigured adapter Subscription service and the TIBCO ActiveMatrix BusinessWorks process. For details on dat files, see Working with dat files in TIBCO Designer on page 4.
inputdata.txt	This file (instead of a database) contains the data that will be used by the server service.
make.bat	Contains commands required to get the Tuxedo service running on Windows (Sets the required environment variables, creates executables, loads the ubbconfig file and boots the Tuxedo service)
make.sh	Contains commands required to get the Tuxedo service running on UNIX (Sets the required environment variables, creates executables, loads the ubbconfig file and boots the Tuxedo service). This script should be run in a k-shell.
server.c	Source file for Tuxedo Server containing the service PRODINFO
ubbprod	ubbconfig file

Setup the Example

Task A Import the project and save as multi-file project

In TIBCO Designer:

1. In the initial dialog box click **Open New Project** and specify a name for the project.
2. Click **Project>Import Full Project**.
3. Click the Zip Archive tab and browse to locate the `Product_RPC_Async.zip` file in the following path,

   TIBCO_HOME`\adapter\adtuxedo\`*version_number*`\examples\Asynchronous\Product\BusinessWorks`
4. Save as a multi-file project.
5. Exit TIBCO Designer.

Task B Start EMS Server

This example uses JMS as the transport type. Ensure that the TIBCO Enterprise Message Service server is running and accessible to the machine on which the adapter is installed before running any of the processes.

Run the `tibemsd.exe` file from your *TIBCO_HOME*`/EMS/bin` directory.

Task C Start the Tuxedo service

Bring up the sample Tuxedo services given in *TIBCO_HOME*`\adapter\adtuxedo\`*version_number*`\examples\Asynchronous\Product` folder, so that the adapter can communicate with them. See Tasks for Preparing Tuxedo on page 91 for details.

Run and Test the Example

Perform the following tasks to test the example.

Task A Run the Adapter in TIBCO Designer

1. Go to **Tool-> Show Adapter Tester**.
2. In the adapter tester window, click on the instance and in the Run Settings tab give the following parameters:

- `Working Directory`: Specify a directory of your choice
- `Adapter Executable`: Select the adapter .exe file (`adtuxedo_native.exe` or `adtuxedo_wrkstn.exe`)

3. Click **Apply** and then **Start**.

Task B Run the TIBCO ActiveMatrix BusinessWorksProcess in TIBCO Designer

1. From the project panel, select the process you want to test. In this example the process is `rpc`.
2. Click the **`Set Breakpoints`** icon.
3. In the window that appears, choose **Select All**, then click **OK**.
4. Click the Tester tab to the left of the project panel. The test panel replaces the project tree.
5. Click the **Start testing viewed process** button.
6. In the process selection window that appears, the `rpc` process is selected by default. Click **Start Selected**. The process is now in Test mode. Any change to the input XML file starts the process.
7. Enter or change the values for `Product ID` and `ServiceName` in the `Get_input.xml` as shown below:

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

<Products
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <Product>

    <PRODUCTID>1</PRODUCTID>

    <ServiceName>PRODINFO</ServiceName>
```

```
    </Product>
</Products>
```

8. As soon as you make changes to the `Get_input.xml` file, the TIBCO ActiveMatrix BusinessWorks process, `rpc` gets triggered.
9. Once the process starter is highlighted (indicating a process has started), click the **Step to next activity** icon to step through the process.
10. The TIBCO ActiveMatrix BusinessWorks process, `rpc` then writes the data received from the Tuxedo service into the `Get_output.xml`.
11. Click the **Stop Testing** icon to return to design mode.

Expected Results

Results for this example will be written into `Get_output.xml` file located in the `D:\examples\Asynchronous\Product\BusinessWorks` folder. A sample output is given below.

```
<?xml version="1.0" encoding="UTF-8"?>
<Product>
    <RequestResponseReplySchema>
        <RequestResponseServiceReply_Product_List>
            <PRODUCTID>1</PRODUCTID>
            <CATEGORYID>234</CATEGORYID>
            <PRODUCTNAME>XYZ</PRODUCTNAME>
            <DESCRIPTION>Books</DESCRIPTION>
            <UOM>12.000000</UOM>
            <PRICE>2.000000</PRICE>
        </RequestResponseServiceReply_Product_List>
    </RequestResponseReplySchema>
</Product>
```

Chapter 10 Request-Response Invocation Service for Rendezvous Adapter Agent-Based Communication

This chapter contains a simple example that demonstrates how the adapter's Request-Response Invocation service works using Rendezvous Adapter Agent-based communication with `FML32` buffers. It also demonstrates the interaction between TIBCO ActiveMatrix BusinessWorks and the adapter in the Request-Response Invocation scenario.

This example uses TIBCO Rendezvous (RV) as its transport type between the adapter and TIBCO environment.

Topics

- *Example Description, page 76*
- *Setup the Example, page 78*
- *Run and Test the Example, page 79*

Example Description

This example simulates the Rendezvous Adapter Agent-based communication that is supported by the adapter Request-Response Invocation service. The Tuxedo service, based on the Sales Order ID that it received from the data sent by the invoking client, returns the DESC, PRODID, UOM, ORDERDATE, QTY, STATUS corresponding to that Sales Order ID.

Following are the flow of events:

1. User sends request from the Tuxedo client.
2. The client invokes the Tuxedo service.
3. The Tuxedo service sends a request to the adapter through the adapter agent.
4. The agent posts the data to the adapter.
5. The adapter puts the request over JMS transport. TIBCO ActiveMatrix BusinessWorks subscribes to the request and returns a response.
6. On getting the response from TIBCO ActiveMatrix BusinessWorks, the adapter forwards the data to invoking the Tuxedo service through the agent.

Location of the Example Files

Files provided with this example are given in the next table. The files are located in the following locations:

TIBCO_HOME\adapter\adtuxedo*version_number*\examples\AgentBased\Sales Order

TIBCO_HOME\adapter\adtuxedo*version_number*\examples\AgentBased\Sales Order\BusinessWorks

File Name	Purpose
Get_input.xml	File containing request data in XML format.
Get_input_parserXsd.xsd	XSD containing input schema definition.
SalesOrderDetails.h	Tuxedo header file containing the FML32 definition for the request buffer.

File Name	Purpose
SalesOrder_Client_Agent.zip	Project zip that contains the preconfigured adapter Subscription service and the TIBCO ActiveMatrix BusinessWorks process. For details on dat files, see Working with dat files in TIBCO Designer on page 4.
adapteragent.h	This is required to compile the Tuxedo server with the adapter agent.
client.c	Source file for the Tuxedo client. Invokes the Tuxedo Service ORDERINFO.
inputdata.txt	This file (instead of a database) contains the data that will be used by the server service.
make.bat	Contains commands required to get the Tuxedo service running on Windows (Sets the required environment variables, creates executables, loads the ubbconfig file and boots the Tuxedo service)
make.sh	Contains commands required to get the Tuxedo service running on UNIX (Sets the required environment variables, creates executables, loads the ubbconfig file and boots the Tuxedo service). This script should be run in a k-shell.
server.c	Source file for Tuxedo server containing the service ORDERINFO.
Ubb	ubbconfig file

Setup the Example

Task A Import the project and save as multi-file project

In TIBCO Designer:

1. In the initial dialog box click **Open New Project** and specify a name for the project.
2. Click **Project>Import Full Project**.
3. Click the Zip Archive tab and browse to locate the `SalesOrder_Client_Agent.zip` file in the following path,

   TIBCO_HOME`\adapter\adtuxedo\`*version_number*`\examples\AgentBased\SalesOrder\BusinessWorks`
4. Save as a multi-file project.
5. Exit TIBCO Designer.

Task B Start the Tuxedo service

Bring up the sample Tuxedo services given in *TIBCO_HOME*`\adapter\adtuxedo\`*version_number*`\examples\AgentBased\SalesOrder` folder, so that the adapter can communicate with them. See Tasks for Preparing Tuxedo on page 91 for details.

Run and Test the Example

Perform the following tasks to test the example.

Task A Run the Adapter in TIBCO Designer

1. Go to **Tool-> Show Adapter Tester**.
2. In the adapter tester window, click on the instance and in the Run Settings tab give the following parameters:

- `Working Directory`: Specify a directory of your choice
- `Adapter Executable`: Select the adapter .exe file (`adtuxedo_native.exe` or `adtuxedo_wrkstn.exe`)

3. Click **Apply** and then **Start**.

Task B Run the TIBCO ActiveMatrix BusinessWorks Process in TIBCO Designer

1. From the project panel, select the `client` process.
2. Click the **`Set Breakpoints`** icon.
3. In the window that appears, choose **Select All**, then click **OK**.
4. Click the **Tester** tab in the project panel. The test panel replaces the project tree.
5. Click the **Start testing viewed process** button.
6. In the process selection window that appears, the `client` process is selected by default. Click **Start Selected**. The process is now in Test mode. Any change to the input XML file starts the process.
7. Enter or change the values for `ORDER_NO`, `PROD_ID_ENTERED`, `DESCRIPTION`, `UNIT_OF_MEASURE`, `ORDER_DATE`, `QTY_ORDERED`, `STATUS` in the `Get_input.xml` as shown below:

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

<SalesOrders
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <SalesOrder>

    <ORDER_NO>1000</ORDER_NO>

    <PROD_ID_ENTERED>12345</PROD_ID_ENTERED>
```

```
    <DESCRIPTION>Books</DESCRIPTION>
    <UNIT_OF_MEASURE>Copies</UNIT_OF_MEASURE>
    <ORDER_DATE>24-Jan-2003</ORDER_DATE>
    <QTY_ORDERED>10000</QTY_ORDERED>
    <STATUS>Active</STATUS>
  </SalesOrder>
</SalesOrders>
```

8. In a command window, start the Tuxedo Client (*TIBCO_HOME*\adapter\adtuxedo*version_number*\examples\AgentBased\SalesOrder\client) and specify the `Order ID` as follows:

   TIBCO_HOME\adapter\adtuxedo*version_number*\examples\AgentBased\SalesOrder>client 1000

9. The client invokes the Tuxedo service. The Tuxedo service sends a request to the adapter through the agent.
10. The agent posts the data to the adapter. The adapter puts the request over RV transport. TIBCO ActiveMatrix BusinessWorks subscribes to the request and returns a response.
11. On getting the response from TIBCO ActiveMatrix BusinessWorks, the adapter forwards the data to invoke the Tuxedo service through the agent.
12. The output is displayed on the client console.
13. Click the **Stop Testing** icon to return to design mode.

Chapter 11 TIBCO ActiveMatrix BusinessWorks: End-to-End Publish-Subscribe for Event Based Communication

This chapter contains a simple end-to-end example that demonstrates how the adapter's Publication and Subscription service works using Tuxedo's Event based communication paradigm with FML32 buffers. It also demonstrates the interaction between TIBCO ActiveMatrix BusinessWorks and the adapter.

This example uses Java Messaging Service (JMS) as its transport type between the adapter and TIBCO environment.

Topics

- *Example Description, page 82*
- *Setup the Example, page 84*
- *Run and Test the Example, page 85*
- *Expected Results, page 87*

Example Description

This example simulates the Event based communication paradigm that is supported by the adapter Publication and Subscription services. The Tuxedo service, based on the Employee ID that it received from the data sent by the invoking client, returns the Department ID, Employee Name and Department Name corresponding to that Employee ID.

Following are the flow of events:

1. The client (`clt` file) calls a Tuxedo service named `WITHDRAWAL`.
2. The Tuxedo server receives the request from the client (`clt` file) processes it and posts an event `BANK_TLR_WITHDRAWAL` along with the message data on the Tuxedo EventBroker.
3. The adapter Publication service, which also registers itself with the Tuxedo EventBroker, receives the message data based on the event name and publishes it.
4. The Subscription service subscribes to the published message and posts the event (`BANK_TLR_WITHDRAWAL_SUBSCRIBE`) with the Tuxedo EventBroker.
5. A second client (client) registers itself with the Tuxedo EventBroker for the event `BANK_TLR_WITHDRAWAL_SUBSCRIBE`.
6. The client receives it and prints the subscribed data on the console. The output parameters are `EMPID`, `DEPTID EMPNAME`, `DEPTNAME`, `DOJ`, `SALARY` and `BASIC`.

Location of the Example Files

Files provided with this example are given in the next table. The files are located in the following locations:

TIBCO_HOME`\adapter\adtuxedo\`*version_number*`\examples\EventBased\Employee`

TIBCO_HOME`\adapter\adtuxedo\`*version_number*`\examples\EventBased\Employee\BusinessWorks`

File Name	Purpose
`EmployeeDetails.h`	Tuxedo header file containing the FML32 definition.

File Name	Purpose
Employee_Pub_Sub.zip	Project zip that contains the adapter subscriber configuration and the TIBCO ActiveMatrix BusinessWorks process. For details on dat files, see Working with dat files in TIBCO Designer on page 4.
client.c	Used to invoke the Tuxedo service named WITHDRAWAL.
eventclient.c	Used to trigger a subscription in Tuxedo.
inputdata.txt	Flat file containing input data.
make.bat	Contains commands required to get the Tuxedo service running on Windows (Sets the required environment variables, creates executables, loads the ubbconfig file and boots the Tuxedo service)
make.sh	Contains commands required to get the Tuxedo service running on UNIX (Sets the required environment variables, creates executables, loads the ubbconfig file and boots the Tuxedo service). This script should be run in a k-shell.
server.c	Source file for Tuxedo server containing the service WITHDRAWAL.
Ubbevent	ubbconfig file

Setup the Example

Task A Import the project and save as multi-file project

In TIBCO Designer:

1. In the initial dialog box click **Open New Project** and specify a name for the project.
2. Click **Project>Import Full Project**.
3. Click the Zip Archive tab and browse to locate the `Employee_Pub_Sub.zip` file in the following path,

   TIBCO_HOME`\adapter\adtuxedo\`*version_number*`\examples\EventBased\Employee\BusinessWorks`
4. Save as a multi-file project.
5. Exit TIBCO Designer.

Task B Start EMS Server

This example uses JMS as the transport type. Ensure that the TIBCO Enterprise Message Service server is running and accessible to the machine on which the adapter is installed before running any of the processes.

Run the `tibemsd.exe` file from your *TIBCO_HOME*`/EMS/bin` directory.

Task C Start the Tuxedo service

Bring up the sample Tuxedo services given in *TIBCO_HOME*`\adapter\adtuxedo\`*version_number*`\examples\EventBased\Employee` folder, so that the adapter can communicate with them. See Tasks for Preparing Tuxedo on page 91 for details.

Run and Test the Example

Perform the following tasks to test the example.

Task A Run the Adapter in TIBCO Designer

1. Go to **Tool-> Show Adapter Tester**.
2. In the adapter tester window, click on the instance and in the Run Settings tab give the following parameters:

- `Working Directory`: Specify a directory of your choice
- `Adapter Executable`: Select the adapter .exe file (`adtuxedo_native.exe` or `adtuxedo_wrkstn.exe`)

3. Click **Apply** and then **Start**.

Task B Run the TIBCO ActiveMatrix BusinessWorks Process in TIBCO Designer

1. From the project panel, select the process you want to test. In this example the process is `pub-sub`.
2. Click the **`Set Breakpoints`** icon.
3. In the window that appears, choose **Select All**, then click **OK**.
4. Click the Tester tab to the left of the project panel. The test panel replaces the project tree.
5. Click the **Start testing viewed process** button.
6. In the process selection window that appears, the `pub-sub` process is selected by default. Click **Start Selected**. The process is now in Test mode.
7. Once the process starter is highlighted (indicating a process has started), click the **Step to next activity** icon to step through the process.
8. Click the **Stop Testing** icon to return to design mode.
9. In a command window, run the client file (*TIBCO_HOME*`\adapter\adtuxedo\`*version_number*`\examples\EventBased\Employee\client`). The client is associated with the Tuxedo EventBroker engine and is ready to subscribe to events based on the event name.
10. Set the following environment variables before running the client file:

- set `TUXDIR=<Enter the home directory of Tuxedo>`

- set `WSNADDR=//<Enter the Host IP Address with the Port for Workstation Listener>`
- set `APPDIR=%TUX_ADAPTER_HOME%\examples\Synchronous\Employee`
- set `PATH=%TUXDIR%\bin;%APPDIR%;%PATH%`

11. In another command window run the `clt` file. (*TIBCO_HOME*`\adapter\adtuxedo\`*version_number*`\examples\EventBased\Employee\clt`)

The environment variables set in step 9 have to be set before running the `clt` file.

Expected Results

On running the `clt` file successfully, the following message is displayed

```
Error = [0]
```

The message is sent and can be seen in the Publication service adapter window After a short delay, it is received in the Subscription service adapter window. The message is displayed in the TIBCO ActiveEnterprise format. The values received in the Subscription service adapter window are as follows:

```
Event Name  = [BANK_TLR_WITHDRAWAL_SUBSCRIBE]

Datatype is long and Value of Emp Id = [1001]

Datatype is long  Value of Dept Id  = [12]

Datatype is string and Value of Emp Name = [Henry Lark]

Datatype is string Value of Dept Name = [Enterprise solutions]

Datatype is string and Value of Join Date = [18-Jan-2003]

Datatype is float and Value of Salary = [2000.000000]

Datatype is Double and Value of Basic = [3000.000000]
```

Appendix A Tasks for Preparing Tuxedo

Before you start the adapter you need to carry out a set of tasks to allow the adapter to access Tuxedo services. This appendix describes these tasks.

Topics

- *Task A: Edit ubbconfig File, page 92*
- *Task B: Edit the Make Script, page 93*
- *Task C: Run the Make Script, page 94*

Task A: Edit ubbconfig File

Edit the `ubbconfig` file, to reflect the same values for the environment variables set on the machine with the Tuxedo client and server installation. The `ubbconfig` file contains default values as bracketed items. Change the default values as per your requirements.

Machines Section

- Change the machine name.
- `TUXDIR`- Tuxedo installation directory.
- `APPDIR`- Directory where the service executable files are located.

The value of this environment variable should not exceed 77 characters.

- `TUXCONFIG` - The name of the Tuxedo configuration file to be generated along with the location. The value specified cannot exceed 64 characters, so ensure that your directory structure is such that the path and name together do not exceed 64 characters.

The value of this environment variable should not exceed 64 characters.

- `ULOGPFX` - The Location of Application Log File.
- `MAXWSCLIENTS` - The maximum number of clients that can connect to Tuxedo. The default is set to 0 (zero). You must set this to 1 or higher in order to boot the Tuxedo services successfully. The maximum number of clients that can be specified is 32767.

Servers Section

- `CLOPT` (modify the IP address to reflect the value specified for the WSNADDR environment variable)
- `MAXWSCLIENTS` is set to 1 or higher.

Refer to Tuxedo documentation for further details on editing the `ubbconfig` file.

Task B: Edit the Make Script

Edit the `make.bat` file, to reflect the same values for the environment variables set on the machine with the Tuxedo client and server installation. The `make.bat` file contains default values as bracketed items. Change the default values as per your requirements. The following values need to be specified

- `TUXDIR` - The Tuxedo installation directory. For example, `TUXDIR=C:\bea\tuxedo8.0`
- `APPDIR` - Directory where the Tuxedo service executable files are located. For example, `APPDIR=`*`TIBCO_HOME`*`\adapter\adtuxedo\`*`version_number`*`\examples\EventBased\Employee`
- `PATH` - The environment variable used to find executables. For example, `%TUXDIR%\bin;%APPDIR%;%PATH%`
- `WSNADDR` - The IP address of the WSL (Workstation Listener) that the client will contact, along with any free port number. This value must be reflected in the `ubbconfig` file. For example, `WSNADDR=//192.168.213.63:4020 (//IP address:Port number)`
- `TUXCONFIG` - The name of the Tuxedo configuration file to be generated along with the location. The value specified cannot exceed 64 characters, so ensure that your directory structure is such that the path and name together do not exceed 64 characters. For example, `%APPDIR%/tuxconfig`
- `ULOGPFX` - The location of Application Log File. For example, `%APPDIR%/ULOG`

Verify that `PATH` on Microsoft Windows, `LD_LIBRARY_PATH` on Solaris and `SHLIB_PATH` on HP-UX contains the location of the adapter agent library (`.dll` on Win 32, `.so` on Solaris and `.sl` on HP-UX).

- `TLOGDEVICE` - Device for binary file that gives the Oracle Tuxedo system all its information. This should be specified for Queue-based examples.
- `QMCONFIG` - The QMCONFIG variable points to an existing device where the UDL either resides or will reside. This should be specified for Queue-based examples.

Task C: Run the Make Script

Run the `make.bat` file. It does the following:

- Converts the `ubbconfig` file into a Tuxedo configuration file. Once the Tuxedo configuration file has been generated, it will be placed in the path specified for the `TUXCONFIG` environment variable.
- The `buildclient` and `buildserver` commands in the file will create executables for Tuxedo client and server programs.
- The `tmboot -y` command in the `make.bat` will boot the services.

For more details, refer to the chapter, *Preparing Tuxedo* in the *TIBCO Adapter for Tuxedo User's Guide*.

Index

C

customer support xvii

E

ENV_HOME xiv

O

Other TIBCO Product Documentation xii

S

support, contacting xvii

T

Task A Create_SalesOrder and Get_SalesOrder Setup 24, 30, 37, 45, 63, 78
Task B Start JMS Server 24, 30, 37, 45, 55, 63, 70, 78, 84
technical support xvii
Third-Party Documentation xiii
TIBCO ActiveMatrix Adapter for Tuxedo Documentation xii
TIBCO_HOME xiv
Typographical Conventions xiv

U

Using TIBCO BusinessWorks 3
Using TIBCO IntegrationManager 4

W

Working with dat files in TIBCO Designer 4