TIBCO ActiveMatrix® Adapter Service Engine for Files

Examples

Software Release 6.0
April 2009

Important Information

Contents

Preface

TIBCO ActiveMatrix Adapter Service Engine for Files comes with preconfigured examples. This manual describes how to run these examples, and explains the configuration of each.

Topics

Related Documentation

This section lists documentation resources you may find useful.

TIBCO ActiveMatrix Adapter Service Engine for Files Documentation

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

- *TIBCO ActiveMatrix Adapter Service Engine for Files Installation* — Read this manual to learn how to install TIBCO ActiveMatrix Adapter Service Engine for Files.
- *TIBCO ActiveMatrix Adapter Service Engine for Files Configuration and Deployment* — This manual explains how to create and configure adapter projects. Information on deploying adapter projects is also included.
- *TIBCO ActiveMatrix Adapter Service Engine for Files Examples* — Read this manual to work through the examples provided with the adapter service engine.
- *TIBCO ActiveMatrix Adapter Service Engine for Files Release Notes* — Read this document for information about new features, deprecated features, and open and closed issues.

Before TIBCO ActiveMatrix Adapter Service Engine for Files can be installed and used, you have to install TIBCO ActiveMatrix Adapter for Files. The following documents form the TIBCO ActiveMatrix Adapter for Files documentation set:

- *TIBCO ActiveMatrix Adapter for Files Concepts* — Read this manual to gain an understanding of adapters in general that you can apply to the various tasks you may undertake.
- *TIBCO ActiveMatrix Adapter for Files Installaltion* — Read this manual to learn how to install TIBCO ActiveMatrix Adapter for Files.
- *TIBCO ActiveMatrix Adapter for Files Configuration and Deployment* — This manual explains how to create and configure adapter projects. Information on deploying adapter projects is also included.
- *TIBCO ActiveMatrix Adapter for Files Examples* — Read this manual to work through the examples provided with the adapter.
- *TIBCO ActiveMatrix Adapter for Files Release Notes* — Read this document for information about new features, deprecated features, and open and closed issues.

Other TIBCO Product Documentation

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

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

Typographical Conventions

The following typographical conventions are used in this manual.

Table 1 General Typographical Conventions

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

Table 1 General Typographical Conventions (Cont'd)

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

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 lists the prerequisite steps to complete before starting the adapter examples.

Topics

Before Starting

Before starting the preconfigured examples, ensure that all required software has been installed and is operating correctly. For a list of required software, see *TIBCO ActiveMatrix Adapter Service Engine for Files Installation Guide*.

The preconfigured examples are located in the *AMX_HOME*`\extensions\adfilesse\samples` directory. The directory contains executable files for Windows and Unix platforms. This guide describes how to run the examples on a Microsoft Windows platform

The *TIBCO_HOME*`\adapter\adfiles\`*version_num*`\examples\reader` and *TIBCO_HOME*`\adapter\adfiles\`*version_num*`\examples\writer` directories contain the directories used in the examples where input files exist and output files are placed. The `reader` directory is used by the Publication service. The `writer` directory is used by the Subscription service.

Examples List

You are encouraged to run the examples, then open the project in TIBCO Business Studio to view the configuration settings. The following projects are available:

- The `FilesAdapterSOASample` project contains the configuration information for the following examples:
  - JMS Service Example
  - SOAP Service Example
  - TIBCO ActiveMatrix BusinessWorks Service Engine Example
  - Mediation Flow Example

  The service assembly archives for the examples are located in the `Deployment Packages` directory of this project.

- The `FilesAdapterEclipseUISample` project contains the configuration information for the following adapter configurations:
  - `delimitedReader_adfilesmodel`

    This adapter instance is configured with a Publication service. The `delimitedReader.wsdl` file that contains the information for this instance is saved in the `FilesAdapterSOASample` project.
  - `delimitedWriter_adfilesmodel`

    This adapter instance is configured with a Subscription service. The `delimitedWriter.wsdl` file that contains the information for this instance is saved in the `FilesAdapterSOASample` project.

  The `delimitedReader.wsdl` and `delimitedWriter.wsdl files` are used in the examples in the `FilesAdapterSOASample` project.
- The `FilesAdapterBWSEDesignerProject` project contains a TIBCO ActiveMatrix BusinessWorks process used in TIBCO ActiveMatrix BusinessWorks Service Engine Example.
- The `com.tibco.sample.java.jms.subscriber` java project contains the java component information used in JMS Service Example.

Using TIBCO ActiveMatrix BusinessWorks Service Engine

TIBCO ActiveMatrix BusinessWorks Service Engine is used in the TIBCO ActiveMatrix BusinessWorks Service Engine Example example. You need to install TIBCO ActiveMatrix BusinessWorks Service Engine before deploying and running the example. See TTIBCO ActiveMatrix BusinessWorks Service Engine documentation for details about installing TIBCO ActiveMatrix BusinessWorks Service Engine.

Using TIBCO ActiveMatrix Administrator

TIBCO ActiveMatrix Administrator is installed with TIBCO ActiveMatrix Service Grid and is used to deploy and run the examples.

Before deploying and running the examples, ensure that you have set up and configured TIBCO ActiveMatrix Administrator correctly. See TIBCO ActiveMatrix Service Grid documentation for details.

Importing the Projects into TIBCO Business Studio

To view the configuration of the examples, you need to decompress the `Files_Adapter_Service_Engine_Samples.zip` file to your workspace and import the projects to TIBCO Business Studio. To do this, follow these steps:

1. Start TIBCO Business Studio.
2. From the File menu, select **Import**. The Import dialog displays.
3. Expand the **General** tree, select **Existing Projects into Workspace**, and Click **Next**.
4. Click **Browse** to select the root directory of the workspace where the projects are located.
5. Select the projects that you want to import to your workspace and click **Finish**. The projects are imported.

Chapter 2 JMS Service Example

Topics

Example Description

This example shows how a Publication service sends messages to two subscribers through the JMS service. In this example, the Publication service parses the `delimited.txt` file and sends its contents in a series of messages repeating every one minute to the JMS service. The JMS service transfers the messages to a topic. The topic then publishes the messages to its subscribers. One of the subscribers is a subscription service. The subscription service receives all the messages and formats, and then writes them to an output file located in a working directory. One minute after opening the file, the service moves the file to its output directory. Each time the publication service is running, the output file is replaced in the subscription service output directory.

The other subscriber is a Java Application component. The Java component receives the messages and prints them to the console. The the java implementation is stored in the `com.tibco.sample.java.jms.subscriber java` project.

The service assembly archives for this example are the `FilesAdapterDelimitedReaderJMSService.zip` and `FilesAdapterDelimitedWriterJMSService.zip` files. You can deploy them using TIBCO ActiveMatrix Administrator.

Deploying and Running the Example

To deploy and run the example, follow these steps:

1. Start HSQLDB Server, Management Daemon, and EMS server.
2. Start the TIBCO ActiveMatrix Administrator server that you created. For example, navigate to the *AMX_HOME*\amx\amxadministrator\2.1\bin directory, execute `amx_admin.bat` to run the server named `admin`.
3. Start the TIBCO ActiveMatrix Administrator server GUI.
4. Start the node that you created.
5. Upload the service assembly archives of the example.
   a. Select the Deploy to an Environment perspective after starting the TIBCO ActiveMatrix Administrator server GUI.
   b. Click the **Upload Service Assembly** button. The Browse to Upload Service Assembly Archive File dialog appears.
   c. Navigate to the archives using the Browse button and specify the name. For example, type **DRJMS** in the Name field.
   d. Click **OK**. The service assembly archive is uploaded.
   e. Repeat step b through step d to upload the other service assembly of this example.
6. Map the service units which the service assembly archives contain to the node that you defined and installed. The Shared Resources for the service units are created.
7. Select the **Configure an Environment** perspective and install the Shared Resources.
8. Select the service assembly archives and click the **Deploy** button to deploy the archives.
9. Start or stop the service assembly archives by clicking the **Start** or **Stop** button.

Results

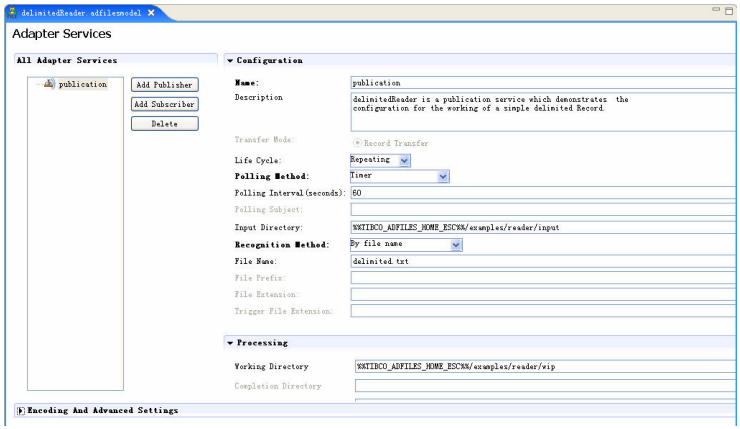
Records in the input file are separated by commas. Records in the output file have been formatted to use semicolons as separators. `Item` lines in the input file were positioned ahead of the `Customer` lines. In the output file, the `Customer` line is positioned ahead of the `Item` lines. See the output of the java application component in the node console.

Input file	Output file
Order, ID41678, 20May2000 Item, GigaWidget, 60, $75 Item, MegaBucket, 48, $125 Customer, Hopkins Associates, ID26490 Order, ID41680, 20May2000 Item, Rt.Clopper, 40, $100 Item, Lt.Clopper, 50, $100 Customer, Jersey WebInovaters, ID46786	Order;ID41678;20May2000 Customer;Hopkins Associates;ID26490 Item;GigaWidget;+6.000000E+001;$75 Item;MegaBucket;+4.800000E+001;$125 Order;ID41680;20May2000 Customer;Jersey WebInovaters;ID46786 Item;Rt.Clopper;+4.000000E+001;$100 Item;Lt.Clopper;+5.000000E+001;$100

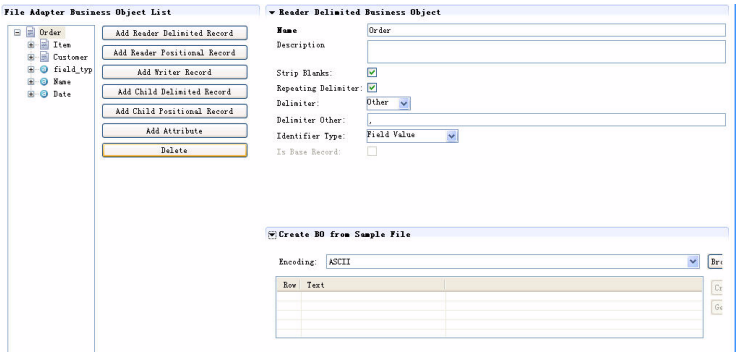
Understanding the Configuration

Publication Service

To enable the publication service to parse the `delimited.txt` file, the service has been configured as shown in the next diagram. The Input Directory, Recognition Method, and File Name fields indicate the directory that contains the input file, how to pick the file, and the file name.



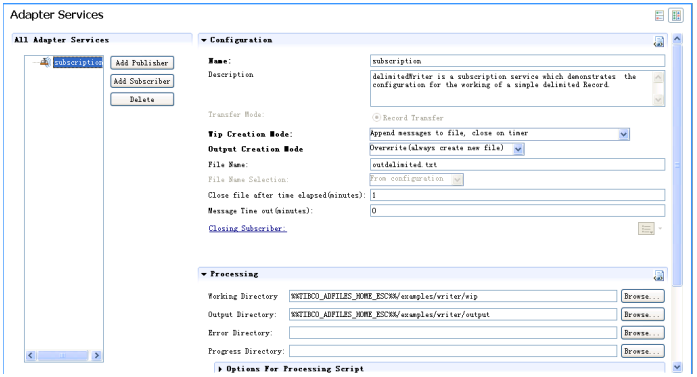
Three delimited file records have been defined for parsing each type of line found in the `delimited.txt` file: `Order`, `Item` and `Customer`. The following diagram shows the definition for the Order file record. The Strip Blanks and Repeating checkboxes are checked so blank spaces between fields on the order line are removed. If any repeated delimiter characters exist between fields, the repeated characters are ignored. The line is identified by a field value. The value is identified under Constant field.



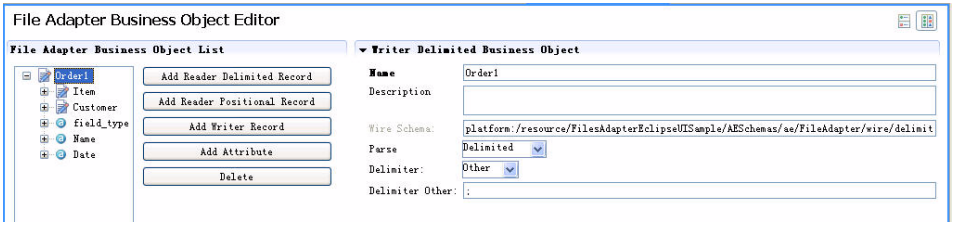
The parent line and child lines to be published are identified in a read Business Object. The read Business Object indicates that the `Order` file record represents the parent line and the `Item` and `Customer` file records represent child lines. The read Business Object is associated with the publication service.

Subscription Service

To enable the subscription service to receive and format messages, the service has been configured as shown in the next diagram. **Append the messages to file, close on timer** is selected in the Wip Creation Mode drop down list.



A write Business Object is defined for the service to include the ActiveEnterprise classes created by the publication service. After the Business Object is defined, write file records are automatically created, one for each class found by the write Business Object. The write file records are arranged so that the `Customer` line will be written below the `Item` line.



The `Customer` write file record is modified so that a tab will be used as the delimiter character. The `Item` and `Order` write file records have been modified to use a semicolon as the delimiter character. The write Business Object is associated with the subscription service.

Using JMS Service

This section describes how the JMS service is used in this example. The following procedure are the general steps for configuring JMS Service binding in TIBCO ActiveMatrix Service Grid. See TIBCO ActiveMatrix Documentation for the details about using the JMS service and developing Java components.

1. Create two composites. In this example, `FilesAdapterDelimitedReaderJMSService.composite` and `FilesAdapterDelimitedWriterJMSService.composite` are created.
2. Create a JMS shared resource. In this example, `Delimited JMS Shared Resource.sharedjmscon` is created.
3. Configure `FilesAdapterDelimitedReaderJMSService.composite`.
   a. Drag and drop the **FilesAdapter** icon from the Components palette into the Components canvas and add the publication endpoint configured in `delimitedReader_adfilesmodel` to the FilesAdapter component. In this example, `DelimitedReaderComponent` is created.
   b. Drag and drop the **JMS** icon from the Services palette into the Services canvas to create a JMS service. In this example, `DelimitedReaderJMSService` is created.
   c. In the Properties view for `DelimitedReaderJMSService`, click the **Target** tab and connect `DelimitedReaderComponent` to `DelimitedReaderJMSService`.

      Click the Binding tab, and type the destination name in the Target Destination Name field. In this example, the destination name is `Delimited.JMSService.Example`.
   d. In the Properties view for the composite, click the **Shared Resource Profiles** tab and add the shared resource that you created in step 2 by clicking the Name column.
4. Configure `FilesAdapterDelimitedWriterJMSService.composite`.
   a. Drag and drop the **FilesAdapter** icon from the Components palette into the Components canvas and add the subscription endpoint configured in

delimitedWriter_adfilesmodel to the FilesAdapter component. In this example, DelimitedWriterComponent is created.

b. Drag and drop the **JMS** icon from the Services palette into the Services canvas to create a JMS service. In this example, DelimitedWriterJMSService is created.

c. In the Properties view for DelimitedWriterJMSService, click the **Binding** tab, and type the destination name in the Target Destination Name field. In this example, the destination name is Delimited.JMSSevice.Example.

d. In the Properties view for the composite, click the Shared Resource Profiles tab and add the shared resource that you created in step 2 by clicking the Name column.

e. Drag and drop the **Topic** icon from the References palette into the Components canvas. In this example, TopicDefinition is created. In the Properties view, click the **General** tab, and specify the Port Type field using the Browse button to add the subscription service configured in .delimitedWriter_adfilesmodel.

f. Drag and drop the **Java** icon from the Components palette into the Components canvas. In this example, JavaSubscriptionComponent is created. In the Properties view, click the **Services** tab, add the subscription service configured in delimitedWriter_adfilesmodel to the component.

   The Java component implementation containing the references code is generated by TIBCO Business Studio automatically. You can specify the Java component implementation parameters. In this example, the com.tibco.sample.java.jms.subscriber project is created.

g. Connect DelimitedWriterJMSService to TopicComponent.

h. Connect TopicComponent to DelimitedWriterComponent.

i. Connect TopicComponent to JavaSubscriptionComponent.

Chapter 3 SOAP Service Example

Topics

Example Description

This example shows how a Publication service sends messages to a subscription service through a SOAP service. The publisher parses the `delimited.txt` file and sends its contents in a series of messages, repeating every minute as SOAP requests to the SOAP service associated with the subscriber. The subscriber receives all the messages and formats, and then writes them to an output file located in a working directory. One minute after opening the file, the service moves the file to its output directory. See Understanding the Configuration to understand the configuration of the services. Refer to Results for the expected results of this example.

The service assembly archives for this example are the `FilesAdapterDelimitedReaderSOAPReference.zip and FilesAdapterDelimitedWriterSOAPService.zip` files. You can deploy them using TIBCO ActiveMatrix Administrator.

Deploying and Running the Example

To deploy and run the example, follow these steps:

1. Start HSQLDB Server, Management Daemon, and EMS server, and the TIBCO ActiveMatrix Administrator server that you created.
2. Start the TIBCO ActiveMatrix Administrator server GUI.
3. Start the node that you created.
4. Upload the service assembly archives of the example.
   a. Select the Deploy to an Environment perspective after starting the TIBCO ActiveMatrix Administrator server GUI.
   b. Click the **Upload Service Assembly** button. The Browse to Upload Service Assembly Archive File dialog appears.
   c. Navigate to the archives using the Browse button and specify the name. For example, type **DRSOAP** in the Name field.
   d. Click **OK**. The service assembly archive is uploaded.
   e. Repeat step b through step d to upload the other archive for this example.
5. Map the service units which the service assembly archives contain to the node that you defined.
6. Select the **Configure an Environment** perspective, and then install the Shared Resources.
7. Select the service assembly archives and click the **Deploy** button to deploy the archives.
8. Start or stop the service assembly archives by clicking the **Start** or **Stop** button.

Start the SOAP service configured with the subscription service, and then start the SOPA service configured with the publication service reference.

Using SOAP Service

This section describes how a SOAP service is used in this example. The following procedure are the general steps for configuring SOAP service binding in TIBCO ActiveMatrix Service Grid. See TIBCO ActiveMatrix Documentation for the details about using the SOAP service and developing Java components.

1. Create two composites. In this example, `FilesAdapterDelimitedWriterSOAPService.composite` and `FilesAdapterDelimitedReaderSOAPReference.composite` are created.
2. Configure `FilesAdapterDelimitedWriterSOAPService.composite`.
   a. Drag and drop the **FilesAdapter** icon from the Components Palette into the Components canvas and add the subscription endpoint configured in `delimitedWriter_adfilesmodel` to the FilesAdapter component.
   b. Drag and drop the SOAP icon from the Services palette into the Services canvas to create a SOAP service.
   c. Connect the SOAP service to FilesAdapter components.
   d. Create an HTTP Server shared resource and specify the port number.
   e. Add the HTTP Server shared resource to the composite.
   f. Generate the WSDL artifact file for the SOAP service by clicking the **Generate WSDL** button in the Binding tab of the SOAP service. The WSDL file is under the Composite tree in the Project Explorer panel. `FilesAdapterDelimitedReaderSOAPReference.composite` will use the generated WSDL file to send SOAP requests.

3. Configure `FilesAdapterDelimitedReaderSOAPReference.composite`.
   a. Drag and drop the **FilesAdapter** icon from the Components Palette into the Components canvas and add the publication endpoint configured in `delimitedReader_adfilesmodel` to the FilesAdapter component.
   b. Create a HTTP Client shared resource and specify the Connect URL. In this example, the URL connected to the HTTP server created in step d is `http://localhost:8333/subscription`.
   c. Drag and drop the **SOAP** icon from the References palette into the References canvas to create a SOAP reference.
   d. Specify the Port Name for the SOAP reference using the Browse button to navigate to the WSDL file created in step f.
   e. Select the SOAP reference. In the Properties view, click the **Binding** tab. In the Transport Configuration area on the right, use the Browse button to select the HTTP Client created in step b.
   f. Create a Java component and add the publication port defined in the `delimitedreader.wsdl file` and the subscription port defined in the `delimitedWriter_subscription_soap.wsdl` file references to the Java component.

      The Java component implementation containing the references code is generated by TIBCO Business Studio automatically. You can specify the Java component implementation parameters. In this example, the `com.tibco.sample.java.soap` project and the `SOAPComponent` class are created.
   g. Connect the Java component to the File Adapter component created in step a.
   h. Connect the Java component to the SOAP reference created in step c.
   i. Double click the Java component to open the source file and add the following code to the `OrderOperation` method: `Reference2.orderOperation(data)`.

      This java code transfers the messages from the publication service to the SOAP reference.

Chapter 4

TIBCO ActiveMatrix BusinessWorks Service Engine Example

Topics

Example Description

This example shows how to use TIBCO ActiveMatrix BusinessWorks Service Engine with TIBCO ActiveMatrix Adapter Service Engine for Files. The publisher parses the `delimited.txt` file and sends its contents in a series of messages, repeating every minute to the BusinessWorks component. TIBCO ActiveMatrix BusinessWorks Service Engine will load the TIBCO ActiveMatrix BusinessWorks process. The process of this example is in the `FilesAdapterBWSEDesignerProject` project. The process has the following components:

- `Receive Partner Notification`

  The Receive Partner Notification activity waits for messages from the publication service, and then transfers them to the next activity, which is the Invoke Partner activity in this process.

- `Invoke Partner`

  The Invoke Partner activity receives the messages and publishes the messages to the subscription service.

The subscriber receives all the messages and formats, and then writes them to an output file. Refer to Understanding the Configuration to understand the configuration of the services. Refer to Results for the expected results of this example.

The service assembly archive for this example is the `FilesAdapterDelimitedBWSESample.zip` file. You can deploy them directly using TIBCO ActiveMatrix Administrator.

Deploying and Running the Example

To deploy and run the example, follow these steps:

1. Start HSQLDB Server, Management Daemon, and EMS server, and the TIBCO ActiveMatrix Administrator server that you created.
2. Start the TIBCO ActiveMatrix Administrator server GUI.
3. Start the node that you created.
4. Upload the service assembly archives of the example.
    a. Select the Deploy to an Environment perspective after starting the TIBCO ActiveMatrix Administrator server GUI.
    b. Click the **Upload Service Assembly** button. The Browse to Upload Service Assembly Archive File dialog appears.
    c. Navigate to the archives using the Browse button and specify the name. For example, type **`DRBWSE`** in the Name field.
    d. Click **OK**. The service assembly archive is uploaded.
5. Map the service units which the service assembly archives contain to the node that you created.
6. Select the service assembly archive and click the **Deploy** button to deploy the archives.
7. Start or stop the service assembly archive by clicking the **Start** or **Stop** button.

Using TIBCO ActiveMatrix BusinessWorks Service Engine

This section describes how to use TIBCO ActiveMatrix BusinessWorks Service Engine in this example. The following procedure are the general steps for using TIBCO ActiveMatrix BusinessWorks Service Engine binding in TIBCO ActiveMatrix Service Grid. See TIBCO ActiveMatrix and TIBCO ActiveMatrix BusinessWorks Service Engine Documentation for the details about using TIBCO ActiveMatrix BusinessWorks Service Engine.

1. Create a TIBCO Designer project in TIBCO Business Studio. In this example, `FilesAdapterBWSEDesignerProject` is created.
2. Create a composite. In this example, `FilesAdapterDelimitedBWSESample.composite` is created in the `FilesAdapterSOASample` project.
3. Drag and drop the **FilesAdapter** icon from the Components Palette into the Components canvas and specify the name. In this example, the name of the component is `DelimitedReaderComponent`. Add the publication endpoint configured in `delimitedReader_adfilesmodel` to the FilesAdapter component.
4. Drag and drop the **FilesAdapter** icon from the Components Palette into the Components canvas and specify the name. In this example, the name of the component is `DelimitedWriterComponent`. Add the subscription endpoint configured in `delimitedWriter_adfilesmodel` to the FilesAdapter component.
5. Drag and drop the **BusinessWorks** icon from the Components Palette into the Components canvas and specify the name. In the Properties view, click the **Implementation** tab, and use the Browse button to select the BW project created in step 1. An archive is generated in the TIBCO Designer project. In this example, `FilesAdapterSOASample_FilesAdapterDelimitedBWSESample_BWSEComponent` is created.
6. Open the TIBCO Designer project, import the folders where the `delimitedReader.wsdl` and `delimitedWriter.wsdl` files are located. The corresponding Schema folder must also be imported. In this example, the `delimitedReader_genresources`, `delimitedWriter_genresources` and `Schema` folders are imported.
7. Open `FilesAdapterDelimitedBWSESample.composite`. Click the BusinessWorks component. In the Properties view, click the **References** tab, and click the **Add** button to add the publication reference that you imported in step 6. A process is generated in the TIBCO Designer project. In this example, `delimitedReader_publication.process` is generated.

8. Connect the publication reference created in step 7 to the `DelimitedReaderComponent` component created in step 3.
9. In the Properties view of the BusinessWorks component, click the **References** tab. Click the **Add** button to add the subscription reference that you imported in step 6, and then add this reference to the process generated in step 7. Connect it to the `DelimitedWriterComponent` component created in step 4.
10. Open `delimitedReader_publication.process`. Drag and drop the **Receive Partner Notification** icon from the Service Palettes panel into the Editor view. In the Configuration tab, select the partner associated with the publication service in the Partner drop-down list. In this example, the partner is `partner_bwse_1`.
11. From the Service Palettes panel, drag and drop the **Invoke Partner** icon into the Editor view. In the Configuration tab, select the partner associated with the subscription service in the Partner drop-down list. In this example, the partner is `partner_bwse_2`. Specify the input of this activity.
12. Connect the Receive Partner Notification activity to the Invoke Partner activity and map a field-to-field mapping.
13. Ensure that `delimitedReader_publication.process` is added to the archive file created in step 5 and then TIBCO ActiveMatrix BusinessWorks Service Engine can load the process while running.
14. Create a service assembly for `FilesAdapterDelimitedBWSESample.composite`.

Each service unit should contain only one component. If you find a service unit that contains more than one component, create new service units and drag the components to the corresponding service units.

Chapter 5 Mediation Flow Example

Topics

Example Description

This example shows how Mediation Flow is used with TIBCO ActiveMatrix Adapter Service Engine for Files. The subscription service is associated with the Mediation Flow component. The Mediation Flow component exposes the subscription service as a SOAP service. The publisher parses the `delimited.txt` file and sends its contents in a series of messages, repeating every minute as SOAP requests. The subscriber receives all the messages and formats, and then writes them to an output file.

A log task is used in the Mediation Flow path. The log task records the information regarding the SOAP requests used in this example.

Refer to Understanding the Configuration to understand the configuration of the services. Refer to Results for the expected results of this example.

The service assembly archives for this example are the `FilesAdapterDelimitedReaderMediationClient.zip and FilesAdapterDelimitedWriterMediationSample.zip` files. You can deploy them using TIBCO ActiveMatrix Administrator.

Deploying and Running the Example

To deploy and run the example, follow these steps:

1. Start HSQLDB Server, Management Daemon, and EMS server, and the TIBCO ActiveMatrix Administrator server that you created.
2. Start the TIBCO ActiveMatrix Administrator server GUI.
3. Start the node that you created.
4. Upload the service assembly archives of the example.
   a. Select the Deploy to an Environment perspective after starting the TIBCO ActiveMatrix Administrator server GUI.
   b. Click the **Upload Service Assembly** button. The Browse to Upload Service Assembly Archive File dialog appears.
   c. Navigate to the archives using the Browse button and specify the name. For example, type **DRME** in the Name field.
   d. Click **OK**. The service assembly archive is uploaded.
   e. Repeat step b through step d to upload the other archive for this example.
5. Map the service units which the service assembly archives contain to the node that you created.
6. Change to Configure an Environment perspective, and then Install the Shared Resources.
7. Select the service assembly archives and click the **Deploy** button to deploy the archives.
8. Start or stop the service assembly archives by clicking the **Start** or **Stop** button.

Start the `FilesAdapterDelimitedWriterMediationSample.zip` archive first.

Using Mediation Flow

This section describes how mediation flow is used in the example. The following procedure are the general steps for configuring mediation flow binding in TIBCO ActiveMatrix Service Grid. See the TIBCO ActiveMatrix Service Grid documentation for details about using mediation flow.

1. Create two composites. In this example, `FilesAdapterDelimitedWriterMediationSample.composite` and `FilesAdapterDelimitedReaderMediationClient.composite` are created.
2. Configure `FilesAdapterDelimitedReaderMediationClient.composite`.

   The steps are the same as configuring `FilesAdapterDelimitedReaderSOAPReference.composite`. The Publication service sends a SOAP request. See step 3 for details.
3. Right click the **Mediation Flows** tree in the Project Explorer panel, select **New > Mediation Flow**. Select the **Simple empty mediation flow** radio button in the Create Mediation Flow dialog and click **Next**. Type the mediation flow name, and click **Finish**. In this example, `MediationSample.mediation` is created.
4. Create an HTTP server shared resource and specify it's port number. In this example, `Delimited Mediation HTTP Server Shared Resource.sharedhttp` is created.
5. Configure `FilesAdapterDelimitedWriterMediationSample.composite`.
   a. Drag and drop the **FilesAdapter** icon from the Components palette into the Components canvas and specify the name. In this example, the name of the component is `DelimitedWriterComponent`. Add the subscription endpoint configured in `delimitedWriter_adfilesmodel` to the FilesAdapter component.
   b. Drag and drop the **Mediation** icon from the Components palette into the Components canvas and specify the name. In this example, the name of the component is `MediationComponent`. In the Properties view, click the

Implementation tab, and use the Browse button to select the mediation flow created in step 3.

c. In the Properties view, click the **Service** tab, and click **Add** to add the subscription service to the mediation component.

d. In the Properties view, click the **References** tab, and click **Add** to add the subscription service to the mediation component as a reference.

e. Connect `DelimitedWriterComponent` to `MediationComponent`.

f. Drag and drop the **SOAP** icon from the Service palette into the Service canvas.

In the Properties view, click the **Target** tab, and select the **MediationComponent>Subscription** radio button to connect the SOAP service to `MediationComponent`.

g. In the Properties view of the Components canvas, click the **Shared Resource Profiles** tab. Add the HTTP server created in step 4 to the composite.

h. Generate a WSDL file for the above SOAP service binding. The WSDL file is used by `FilesAdapterDelimitedReaderMediationClient.composite` to send SOAP requests.

i. Double click **MediationComponent** to open it. Select **Order_operation** in the Mediation Interfaces canvas and select its target operation.

j. Click the **Input** button. An input path appears. Place a Log Task in the input path and select the items to send to the log file.

Index