TIBCO ActiveMatrix® Adapter for Kenan/BP

Configuration and Deployment

Software Release 6.0
November 2009

Important Information

Contents

Figures

Tables

Preface

This document describes how to create, configure, and deploy projects using TIBCO ActiveMatrix Adapter for Kenan/BP.

Topics

Related Documentation

This section lists documentation resources you may find useful.

TIBCO ActiveMatrix Adapter for Kenan/BP Documentation

The following documents form the TIBCO ActiveMatrix Adapter for Kenan/BP documentation set:

- *TIBCO ActiveMatrix Adapter for Kenan/BP Concepts* Read this manual to gain an understanding of the product that you can apply to the various tasks you may undertake.
- *TIBCO ActiveMatrix Adapter for Kenan/BP Installation* Read this manual to learn how to install TIBCO ActiveMatrix Adapter for Kenan/BP.
- *TIBCO ActiveMatrix Adapter for Kenan/BP Configuration and Deployment* Read this manual for instructions on creating, configuring, and deploying adapter projects.
- *TIBCO ActiveMatrix Adapter for Kenan/BP Examples* Read this manual to work through the examples provided with the adapter.
- *TIBCO ActiveMatrix Adapter for Kenan/BP 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. 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 following Kenan/BP documentation:

- *API TS Guide*
- *API TS Reference*

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 book provides information on configuring an adapter instance, using adapter features, and deploying the adapter to the runtime environment.

Topics

Overview

TIBCO ActiveMatrix Adapter for Kenan/BP allows enterprise systems to integrate with Kenan/BP. The adapter mediates between Kenan/BP and the TIBCO environment. The adapter allows external applications to send and receive messages to and from Kenan/BP.

The adapter maintains data integrity in both directions, and integrates other software in the TIBCO environment seamlessly into an enterprise.

Features

TIBCO ActiveMatrix Adapter for Kenan/BP supports the following capabilities:

- **PAM Compliance** The adapter performance statistics can be monitored from TIBCO Administrator through its component called PAM (Process Activity Monitoring). Performance statistics methods and other adapter-specific methods are added to the TIBCO Adapter SDK class microagent. Performance statistics can be obtained through these methods.
- **Load Balancing** The adapter can run multiple request-response services in a distributed queue, resulting in load balancing. Load balancing is implemented through the use of the TIBCO Rendezvous distributed queue protocol.
- **Fault Tolerance** The adapter supports fault tolerance.
- **Tracing and Tracking Facility** The adapter's error management uses the TIBCO ActiveEnterprise exception handling and tracing facility. The adapter also incorporates the `Tracking Info` functionality of the TIBCO Adapter SDK.
  - Providing end-to-end traceability of business documents in case of errors or processing that hangs.
  - Helping identify the source of a business document through all the components in a given process.
- **Exception Handling** Exception handling is supported at both the adapter and Kenan/BP levels. If the adapter encounters any exceptions while processing the request, the error message is returned as part of the response schema.
- **Multithreading** The adapter supports a static number of threads. The number of threads is specified at design time. See Multithreading for more information. The number of connections to the application server will not exceed the number of threads. With multiple threads, the adapter processes multiple messages concurrently. The maximum number of threads cannot exceed the minimum PoolableATMI Size (specified in `pools.properties`)

and the MAXWSCLIENTS (specified in `KenanFX.ubb`). Refer to the *Kenan/BP System Administration Guide* for more details.

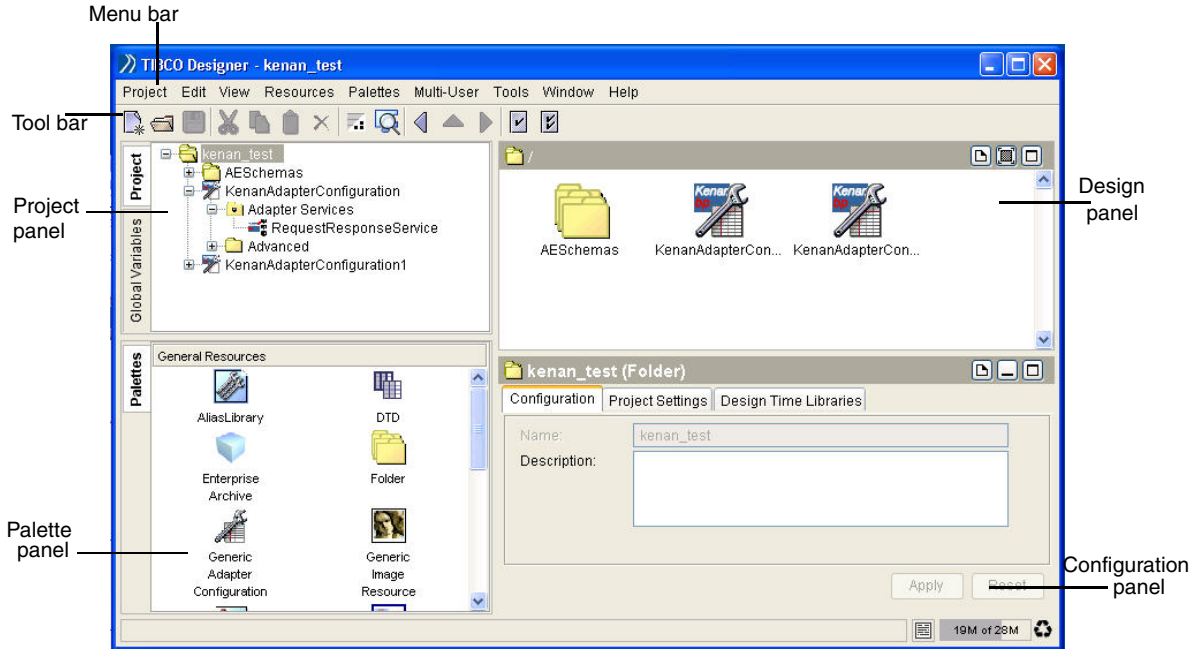
- **Transaction Handling** TIBCO ActiveMatrix Adapter for Kenan/BP provides Transaction support through User Defined Transactions (UDT) and Orders Services. See Chapter 4, Transaction Handling.

Configuration

TIBCO Designer, the design-time component, is an easy-to-use graphical user interface. The TIBCO ActiveMatrix Adapter for Kenan/BP palette in TIBCO Designer is used to create adapter instances, configure adapter services, and import Kenan/BP XSD files into a project.

Before using TIBCO Designer, make sure you read the TIBCO Designer product documentation. The following figure shows the TIBCO Designer interface.

Figure 1 TIBCO Designer main window



Project Panel

Each TIBCO Designer window contains one and only one project, which is represented as the root folder in the panel.

Projects are the key organizational principle for the configuration information you specify. A *project* is a collection of all configured resources. *Resources* are the components of a project. For example, an adapter publication service is a resource. Resources can be complex and contain other resources, much like a folder can contain other folders on your computer file system. Together, these resources make up your integration project. The top-level (root) folder in the project tree panel represents a project. The top-level folder is initially named `Untitled` and is renamed to the name of the project when you save the project for the first time.

Most adapter resources have context-sensitive help available for the configuration of that resource. Right-click on the resource and choose **What Is This?** from the popup menu for more information on configuring the resource.

An adapter project contains the following folders:

- AESchemas Folder

  The *AESchemas* folder is the default location for all TIBCO ActiveEnterprise schema files. Each schema file contains a collection of classes, scalars, associations, unions, and sequences.

- Adapter Services Folder

  The *Adapter Services* folder contains services available to the adapter. Most adapters include publication, subscription, request-response, and request-response invocation services.

- Advanced Folder

  The *Advanced* folder contains resources created by TIBCO Designer while the adapter is configured. For example, each time you add a service to an adapter, a session and endpoint are created and stored in the Advanced folder. Other resources such as advanced logging resources are accessed directly from the folder. Adapter developers typically do not access resources in this folder. Most of the adapter configuration is done by changing resources that are available from the Adapter Services folder.

Palette Panel

Palettes organize resources and allow you to add them into your project. Palettes are available from the palette panel in TIBCO Designer. Resources are visible components in a palette. You select resources in the palette panel and drag-and-drop them into the design panel to add them to your project.

Each adapter you install adds one or more palettes during installation. The palette that is displayed depends on the resource selected in the project tree and on your preferences. In the default view, the current selection in the project tree determines which palettes are displayed in the palette panel.

Design Panel

The design panel displays the current resource selected in the project tree panel. For resources that contain other resources, the contents of the selected resource are shown in the design panel. For example, if you select a folder, its contents are displayed.

Configuration Panel

The configuration panel allows you to specify configuration options for the selected resource. The type and the purpose of the resource determine the contents of the configuration panel. There are usually one or more tabs in the configuration panel that allow you to access the various configuration options. The tabs provide an organization to the options for the resource.

You can click the question mark icon (`?`) in the top right corner of the configuration panel for online help with the current selection.

For each tab, you must click **Apply** after you have specified configuration information before you can select another tab. If you decide you do not want to add the configuration information, click **Reset** before you apply any changes to return to the previous values for each field in the tab.

Projects

A project is a named collection of data, usually schema data, and configuration data that is persistently stored. Each project is opened and saved in multi-file format, which allows the project to be used with a version control system. It allows different developers to collaborate on a project and merge changes as needed.

When a project is ready to be deployed, it can be created or exported in the following formats:

- Enterprise Archive File
- Local Repository
- Server Repository
- ZIP Archive

Enterprise Archive File

An Enterprise Archive file contains information about the adapter instances and processes you want to deploy. The format is used by TIBCO Administrator. The EAR file is imported into Administrator where you can deploy, start, and manage the adapter instance on the machines of your choice.

Local Repository

A project exported to a local repository is saved in `.dat` format. Projects saved in `.dat` format should only be used for development and testing. The format can be used where data is not shared by more than one adapter. It is possible to have a few local adapters accessing a local project in read-only mode. It is, however, not possible to have more than one local adapter accessing a local project in read and write mode.

Data are loaded at startup for local projects, so a local project has higher memory requirements.

Server Repository

A project exported to a server repository is managed by a TIBCO Administration Server running in a separate process, typically elsewhere on the network. One or more adapters can communicate with a project managed by an Administration Server. Each can support multiple projects.

An Administration Server is identified by a name that must be unique among all administration servers on a network. The server-based mode of operation is scalable and generally recommended for production situations. Server repositories allow multiple simultaneous write operations with locking, automatic updates of clients, and notification.

Data are loaded on demand for server-based projects.

ZIP Archive

A project exported to a ZIP archive is written to the location you specify as a read-only ZIP file. A project exported as a ZIP archive can be imported into TIBCO Designer.

Version Control

TIBCO Designer allows multiple developers to work on the same project and to use file sharing (locking) or a version control system so that the same resource is not changed by two developers at the same time. Different users can then add resources to the project and lock the parts of the project they are working on.

TIBCO Designer creates a file that can be shared and locked for each top-level resource, such as an adapter configuration. It does not create a file for each resource. As a result, for example, you can lock an adapter configuration but cannot lock individual adapter services.

When an adapter service is configured, the adapter creates a corresponding set of schema files. A warning is displayed when the files are created advising you to add the files to your version control system. You must add the files to your version control system and ensure they are checked-in, otherwise your project will not be managed correctly by the version control system.

TIBCO Designer also creates folders for folders you create in your project. You can lock each folder as needed.

Deployment

After an integration project has been developed and tested, it is necessary to deploy the runtime components to the machines on which they will ultimately run in a production environment. An adapter instance can be deployed, started and managed using TIBCO Administrator from the Administrator web browser.

Using TIBCO Administrator, you can upload the EAR file, deploy the adapter on the machine(s) of your choice, and set runtime options before deployment. Additionally, you can start and stop the adapter using TIBCO Administrator.

TIBCO Administrator also provides built-in tools to monitor and manage the adapter.

TIBCO Administrator provides user, resource, and application management modules for adapters.

- **User Management.** This module allows you to set permissions for adapter users. You define authentication, users and groups, and assign access control lists to users. This includes security for server-based projects at design time and for deployed applications at runtime.
- **Resource Management.** This module allows you to monitor machines and all running applications in a TIBCO administration domain. Alerts can be created, for example, to notify an administrator if the number of processes or the amount of disk usage exceed a certain number.
- **Application Management.** This module allows you to upload Enterprise Archive (EAR) files, and create, configure, and deploy adapters. This console is also used to start and stop adapters.

TIBCO Administration Domain

A TIBCO administration domain is installed only if you have also installed the User Management module.

A *TIBCO administration domain* is a collection of users, machines, and components that an administration server manages. There is only one Administration Server for each administration domain. Components within an administration domain can communicate with systems outside of the domain, but the domain is the administrative boundary of your enterprise integration project.

See the TIBCO Administrator product documentation for more information.

TIBCO Administration Server

The TIBCO Administrator Server provides a central storage and distribution point for configuration data and schema data needed by an adapter. The server is included in both Administrator editions.

Each administration domain has one and only one TIBCO Administration Server. The *TIBCO Administration Server* is the machine process that handles the stored project and requests to manage the TIBCO administration domain.

The TIBCO Administrator Server contains its own web server (Apache Tomcat) that can be accessed via the TIBCO Administrator GUI for configuration and monitoring information.

TIBCO Administrator GUI

You can access the TIBCO Administration Server using the web-based TIBCO Administrator GUI. The GUI allows you to create users and assign access to projects managed by the Administration Server. You can invoke the GUI from any machine in a TIBCO administration domain.

Adapter Project Lifecycle

This section describes the high-level steps required to configure and deploy an adapter. Each of these steps are described in detail in subsequent chapters. Adapter projects are configured using TIBCO Designer.

Configuration

Task A Define an Adapter Project

When starting TIBCO Designer, you create or select a project. A project contains adapter configuration information, such as which service and messaging transport to use, logging options, and other specific settings. A project is opened and saved in muti-file format which allows a version control system to manage the files associated with the project.

Task B Set Global Variables

By default, each project you create in TIBCO Designer includes several global variables. Global variables provide an easy way to set defaults for use throughout your project. Default values are predefined for some of the variables. You can define additional variables and, optionally, set their values when configuring your adapter.

When the project is deployed and the configured adapters are run, all occurrences of the global variable name are replaced with the global variable value.

A global variable value set in TIBCO Designer can be overridden at runtime by redefining the value in TIBCO Administrator.

See Using Global Variables for more information.

Task C Configure an Adapter Service

The adapter supports the request-response service. See Adapter Services Options for more information.

Task D Test the Adapter Configuration

The adapter tester can be used to verify an adapter service after it has been configured. When invoked, all adapter services configured in the project are displayed. You select the adapter service to test, and start and stop the adapter from the tester. The tester window displays adapter output within the tool so you can easily view the results.

TIBCO ActiveMatrix Adapter for Kenan/BP only supports the adapter tester on the supported UNIX platforms.

Deployment

During development, you save your design to a project. When you are ready to deploy your project to a machine, you generate an Enterprise archive file (EAR file) from TIBCO Designer. The EAR file contains information on what you want to deploy.

Task E Generate and Import an Enterprise Archive File

An Enterprise Archive file contains adapter instance configuration information that is used by a runtime adapter. An Enterprise Archive file is generated using TIBCO Designer and imported into TIBCO Administrator.

See Creating an EAR File in TIBCO Designer for more information.

Task F Specify Deployment Information

After importing an Enterprise Archive file, the adapter can be deployed. This involves:

- Assigning adapter services to the machines in the administration domain.
- Specifying startup options for each process engine and adapter service.

See Deploying the Project for more information.

Task G Specify Monitoring Options

Before starting the adapter, you can specify monitoring options, including:

- Specifying alerts or TIBCO Hawk rulebases for each machine.
- Specifying alerts and TIBCO Hawk rulebases for an adapter service.
- Setting log file properties for an adapter service instance.

Task H Start the Adapter

The adapter is started and stopped using the TIBCO Administrator GUI.

See Starting or Stopping the Adapter for more information.

Chapter 2 Getting Started

Topics

Overview

This chapter provides a short exercise that shows you how to configure TIBCO ActiveMatrix Adapter for Kenan/BP with a request-response service. It also shows you how to start and stop the adapter.

Before Starting

Before starting the configuration exercise, ensure that all required software has been installed and is operating correctly. For a list of required software, see *TIBCO ActiveMatrix Adapter for Kenan/BP Installation*.

You should know how to drag and drop icons in TIBCO Designer and be familiar with saving projects. If you are not familiar with these topics, refer to the TIBCO Designer documentation, which is available by clicking **Help>Designer Help** in TIBCO Designer.

Configuring the Adapter Components

This section leads you through a quick configuration exercise. This exercise covers the following tasks:

- Task A, Create a TIBCO Designer Project
- Task B, Create and Configure an Adapter Instance
- Task C, Create and Configure a Request-Response Service

Task A Create a TIBCO Designer Project

TIBCO Designer is used to create projects and configure adapter instances. When starting TIBCO Designer, you must first create or open an existing project.

1. Start TIBCO Designer and select **New Empty Project**.
2. In the Save Project dialog, click the **Browse** button to select the location of the project and click **OK**. A project is created.

Task B Create and Configure an Adapter Instance

An adapter instance can contain one or more request-response services. Options for logging, startup, and monitoring are set on the adapter instance. In this exercise, default values are used for these options.

1. Drag and drop the **KenanAdapterConfiguration** icon from the palettes panel to the design panel. An adapter instance is created and the default instance name is `KenanAdapterConfiguration`. You can change the instance name.
2. Identify the file log options in the Logging tab if you want to log trace messages. You can configure the levels of trace messages you want to log and where trace messages are sent.
3. Click the **Apply** button to save the changes.

Task C Create and Configure a Request-Response Service

1. In the project panel, expand the **KenanAdapterConfiguration** folder, then select the **Adapter Services** folder.
2. Drag and drop the **Request-Response Service** icon from the palette panel to the design panel.
3. In the Configuration tab, select **Rendezvous** or **JMS** in the Transport Type drop-down list.
4. Leave other default values unchanged.

5. Save the project.

Deploying the Adapter

This section describes how to deploy the runtime adapter from Adapter Tester and from a shell on a UNIX platform.

You cannot deploy the runtime adapter on the Microsoft Windows platform. If you configure the adapter on a Microsoft Windows platform, you can either convert the project to a DAT file or convert the project to an EAR file and then and deploy it on a UNIX machine.

Deploying the Runtime Adapter Using Adapter Tester

The adapter tester is a tool that is used for testing runtime adapters. The adapter tester tool is only supported on UNIX. Follow these steps to deploy the adapter instance.

1. Open the project that you have configured in TIBCO Designer.
2. From the Tools menu, select **Show Adapter Tester**.
3. Specify the start directory for the adapter instance in the Run Settings tab separately.
4. Start the adapter instance.

   You can see the progress of the request-response service from the console.

Deploying the Runtime Adapter from a Shell

To deploy the runtime adapter from a shell, follow these steps:

1. Convert the project to a DAT file.
   a. Open the project that you have configured in TIBCO Designer.
   b. From the Project menu, select **Export Full Project**.
   c. Browse and select the location of the directory where you want to save the DAT file in the Export Project dialog, for example *TIBCO_HOME*`/adapter/adkenan/`*version_num*`/test`.
   d. Enter the name for the DAT file in the Project Name field in the Export Project dialog, for example, `rpc_sample`.
   e. Click OK.
   f. The `rpc_sample.dat` file will be generated and saved in *TIBCO_HOME*`/adapter/adkenan/`*version_num*`/test`.

2. Create a TIBCO Runtime Agent file.

   The TIBCO Runtime Agent file is a runtime configuration file with the `tra` suffix.

   The installation program generates the TIBCO Runtime Agent file for TIBCO ActiveMatrix Adapter for Kenan/BP, and is called `adkenan.tra`. On UNIX, if the TIBCO Runtime Agent file generated by the installation program is copied and used by the runtime adapter (with the appropriate modifications), you need to set the required environment for the adapter before running the adapter.

   You can use the `adkenan.tra` file or make a copy of the `adkenan.tra` file and update the related variables.

   a. Navigate to the *TIBCO_HOME*`/adapter/adkenan/`*version_num*`/bin/` directory. Make a copy of the `adkenan.tra` file, and rename it, for example `rpc_sample.tra`.

   b. Update the related variables.

      At a minimum, the following must be provided:

      `tibco.repourl`—the pathname of the TIBCO Designer project or the DAT file of the project

      `tibco.configurl`—the name of the adapter configuration

      `application.args`— the path name of the properties file used by the runtime adapter

      For example, you can use a text editor and set the following properties:

      `tibco.repourl` *TIBCO_HOME*`/adapter/adkenan/`*version_num*`/test/rpc_sample.dat`

      `tibco.configurl KenanAdapterConfiguration1`

      `application.args adkenan --propFile` *TIBCO_HOME*`/adapter/adkenan/`*version_num*`/bin/rpc_sample.tra`

3. Deploy the adapter instance from a shell

   a. Navigate to the *TIBCO_HOME*/tibco/adapter/*version_num*/bin/ directory in a shell.

      If you are using the default `adkenan.tra` properties file to start the adapter, type **`adkenan --run`** in the shell. Because the properties file has the same name prefix as the executable and is in the same folder, it need not be specified in the shell.

      If you are using a custom TRA properties file that is located in the *TIBCO_HOME*/tibco/adapter/*version_num*/bin/ directory, type **`adkenan --run --propFile`** ***filename*`.tra`** in a shell to start the adapter. The

absolute pathname to the properties file must be given if the TRA file is located in a different directory than the executable.

b. Stop the adapter by publishing a message on the `adkenan terminate` subject.

Chapter 3 Adapter Instance Options

This chapter explains how to create an adapter instance by configuring the standard settings. All configuration tasks are performed in TIBCO Designer and the information is stored in a project that is later used by the runtime adapter

Topics

Overview

Please read the following sections before starting to configure an adapter.

- Adapter Instance Tabs
- Adapter Services Options

Configuration Tasks

Use the following sequence to create and configure an adapter.

1. Start TIBCO Designer and open a multi-file project. See the *TIBCO Designer* documentation for details.
2. Drag and drop the **KenanAdapterConfiguration** icon from the palettes panel to the design panel. This creates an adapter instance named, by default, `KenanAdapterConfiguration`.
3. Configure the adapter instance. See Adapter Instance Tabs for more information.
4. Add one or more request-response services to the adapter instance by dragging and dropping the **Request-RespondService** icon from the palettes panel to the design panel.
5. Configure adapter services for the adapter instance. See Adapter Services Options for more information.
6. Save the project.

After configuring the adapter, create the runtime adapter property file and add the project name and adapter instance name.

Multithreading

The adapter will implement multithreading using TIBCO Adapter SDK's `MDispatcher` class. A fixed number of threads will be created by the `MDispatcher` class. The number of threads can be specified at design time in the Multithreading Tab of the adapter instance.

Since multithreading is implemented at the session level, you have to specify the number of threads for each session. A group of services can be multithreaded under the same session.

When the adapter starts up, a session object is created for each session configured under the instance. If the thread count set for the session is more than 0, an equal number of dispatcher objects will be created for that session to dispatch events. If the thread count has not been configured, the default TIBCO Adapter SDK thread will dispatch the events for all sessions.

The adapter does not maintain a connection pool because this is done by the end application itself. Each thread maintains and reuses its own connection object. The connections are refreshed whenever the target application goes down. Each time a request is received, the connection object is used to execute the API call.

Connection Management

This section explains the connection management process of the adapter and its reconnection mechanism.

Reconnection Mechanism

After each request has been processed by the adapter, the request-response service checks for any errors due to the following reasons:

- The Security Server stops working or responding.
- One or more Tuxedo services stop working.
- The Kenan/BP database encounters errors.

If a connection error occurs, the adapter attempts to reconnect to the system for a fixed number of times with a fixed time difference between each retry. These parameters can be configured in the Run-time Connection Tab at design time.

The state of the application (up or down) is checked using the response from the Kenan/BP application. The error message is extracted from the response, based on where the connection retry mechanism is originated.

There is a limitation in the Kenan/BP client libraries where after a Tuxedo service reboot, the used connections can not be reused anymore. As a result, the number of valid connections in the pool decrease and the adapter cannot refresh a connection after the server stops working and is brought back up.

Shutting down the adapter can be delayed by increasing the number of connections in the `pools.properties` file and by using fewer threads (proportional to the connection pool size). For example, if the number of threads configured in the adapter is **`4`** and the connection pool size is **`40`**, the adapter can perform **`10`** reconnection attempts successfully. The connection pool size can be set using the `PoolableATMI` parameter in the `pools.properties` configuration file. Refer to the *Kenan/BP System Administration Guide* for details.

Saving the Project

Configuration information for an adapter and all other parameter settings related to the adapter are saved as a project. At any time while configuring the adapter, you can save the associated project. Each time you save a project, any configuration information you have entered is saved as a project.

For detailed steps and more information about exporting or importing projects to different formats (such as `.dat`), see the TIBCO Designer documentation.

Testing the Adapter

You can use the Adapter Tester to verify that an adapter instance is configured correctly. The tester is invoked from the TIBCO Designer Tools menu and is documented in the TIBCO Designer documentation.

Adapter Instance Tabs

The following tabs are available when configuring an adapter instance.

- Configuration Tab
- Run-time Connection Tab
- Multithreading Tab
- General Tab
- Logging Tab
- Startup Tab
- Monitoring Tab

Configuration Tab

You must define the options in this tab before other options can be configured. Click **Apply** before leaving this dialog to apply the changes.

Table 3 Adapter Instance Configuration

Field	Description
Instance Name	This is the name that was specified when creating the adapter configuration. See Guidelines for Choosing an Instance Name for more information.
Description	A short description of the adapter configuration.
Version	The version string indicates the TIBCO ActiveEnterprise (AE) configuration format in which the adapter instance is saved. An adapter instance can be saved in AE 4.0, AE 5.0 or AE fro1 format. When a new adapter instance is created in TIBCO Designer 5.x, the version string is set to `AE Version 5.1`. When a 4.x adapter instance is opened in Designer 5.x, the `Version` field is set to `AE Version 4.0`. ---If a 4.x adapter instance is to be run against a 4.x runtime adapter, the instance must be saved with the `Version` field set to `AE Version 4.0`. ---If you are using TIBCO Designer 5.x to modify 4.x adapter instances, change only the features supported by the 4.x. runtime adapter and use the validation utility to verify the instance before deploying the project. Invoke the utility from the Project>Validate Project for Deployment menu in Designer. To change versions, click the **Change Version** button.

Table 3 Adapter Instance Configuration

Field	Description
Message Filter	Specify a message filter if you have configured a message filter resource for use with the adapter. The plug-in allows you to manipulate incoming and outgoing data before sending it on the network or handing it to the target application. Plug-ins can be written using TIBCO Adapter SDK. See *TIBCO Adapter SDK Programmer's Guide* for information about writing a message filter.
Show All Tabs	Select this box to display additional tabs for configuring advanced options.

Guidelines for Choosing an Instance Name

- An instance name must use alphanumeric characters. An underscore (_) character can be used. The entire instance name must be less than 80 characters. The space character cannot be used in an instance name.
- An instance name cannot use global variables.
- An instance name must be unique with respect to other adapter instances for the same adapter in the project. The same instance name can be used to name an adapter instance for a different adapter in the same project. For example, a Kenan adapter instance named `TEST` and a Siebel adapter instance named `TEST` can coexist in the same project.
- Each instance name must be unique for each adapter within a project even if each instance is defined in a different folder. In other words, configuring same-named adapter instances in different folders will not make their names unique.

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

Run-time Connection Tab

This tab allows you to define the connection parameters used to connect to the Kenan/BP application during configuration. At design time, the palette connects to the Kenan/BP application to fetch the details for various operations. Specify values in the following fields to set up the runtime connection.

Table 4 Runtime Connection Tab

Field	Description
Security Server Realm	The realm of the Kenan/BP server. Global variables can be used to specify the realm of the Kenan/BP server. This is a mandatory field.
Security Server Login Name	A valid login name used to log on to the Kenan/BP application. Global variables can be used to specify a valid login name. This is a mandatory field.
Security Server Password	A valid password corresponding to the username. Global variables can be used to specify a valid password. This is a mandatory field.
Remember Password	This checkbox is disabled.
Maximum Number of Reconnect Attempts	Specify the total number of reconnection attempts to make before the runtime adapter or adapter service is stopped. A value of `-1` means reconnection attempts will continue indefinitely. The default is `-1`.
Interval Between Reconnect Attempts (milliseconds)	Specify the time interval in milliseconds to elapse between each reconnection attempt. The default value is `100`.
Adapter Termination Criteria	This field is disabled.

Multithreading Tab

You can specify the ThreadCount variable for each session in the Multithreading tab.

Table 5 Multithreading Tab

Field	Description
Session Name	The sessions for which the dispatcher count can be edited are listed.

Table 5 Multithreading Tab

Field	Description
Number Of Threads	Specify the number of threads in this field. The maximum value that can be specified for the number of threads is one less than or equal to the number of threads specified in the property PoolableATMI Size of pools properties: `Number of Threads <=PoolableATMI.Size -1`. The limitation here is that when using the adapter in the mixed mode (in other words, when sending both normal requests and custom call requests), the number of threads configured should be half of the value specified for the PoolableATMI Size of pools properties. Note that by default, the adapter will create one thread to dispatch events for each session listed. The zero in the Number Of Threads column means one thread.

General Tab

This tab allows you to set a termination subject or topic and specify the encoding type, debug level, and verbose mode. Note that the adapter should communicate only with other applications that support the same code pages or Unicode.

Click **Apply** before leaving this dialog to apply the changes.

In the Configuration tab, check the **Show All Tabs** checkbox. Then select the **General** tab. You can configure general options such as Termination Subject and Encoding values which will be used by all services.

Table 6 General Tab

Field	Description
Termination Subject or Topic	A message sent on this subject (if TIBCO Rendezvous is the transport) or topic (if JMS is the transport) stops the adapter. In most cases, you should use the default value. See TIBCO Rendezvous documentation for information about specifying subject names. See the TIBCO Enterprise Message Service documentation for information about publishing on a topic.
Adapter Encoding	Select the encoding from the drop-down list.
Response Namespace	This field allows you to designate the namespace of the response message from the adapter. For example, if you type **`CSG`** in this field, the response message from the adapter will include namespace information which is `xmlns="CSG"`. By default, the value of this field is `CSG`.
Enable DirectParse	Check or uncheck this checkbox to enable or disable the direct parse function for the UDT call. For more information, please refer to Chapter 4, Transaction Handling. By default, this field is not checked.

Logging Tab

Use these settings to configure a log file or log sinks, including which types of trace messages you want to log and where the trace messages are sent. Click **Apply** before leaving this dialog to apply the changes.

Table 7 Logging Tab

Field	Description
Use Advanced Logging	If unchecked, the standard log file is used. This is the default. Fill out the remaining fields on this tab. You do not need to read the rest of this field description. If checked, you can set two standard output destinations (sinks) for trace messages and set the tracing level for the roles selected. The following sink types are available: ---File ---STDIO ---Hawk ---Network See Creating Log Sinks for more information.
Log to Standard I/O	If checked, trace messages are displayed in the command prompt window where the adapter is started. This is the same as creating a STDIO sink. If unchecked, trace messages do not display in the window.
Log File	Specify the name of the log file to which trace messages are written. This is the same as creating a file sink. Global variables can be used to specify the location of the log file. See Using Global Variables for more information. Type the name and file system path, or click **Browse** and select an existing log file. If no file name is specified, trace information is not written to a file.
Log Info/ Debug/ Warning/ Error Messages	Select the types of trace messages you want to log.

By default all the error, warning, debug, and information messages are printed in the console window in which the adapter was started. Alternatively, you can specify a log file and path to redirect trace output to a log file located anywhere on your file system. The default log file name is *%%DirTrace%%/%%Deployment%%.%%InstanceId%%*`.log`, and is saved in the same directory where your project (repository instance) is stored.

Most errors received by the adapter are logged. The only errors that might not be logged are any TIBCO Rendezvous or TIBCO Adapter SDK errors that appear at startup time before tracing can be initialized.

Logging trace messages is helpful for troubleshooting. There are four levels of trace messages that you can log: *Information*, *Warning*, *Debug*, and *Error*. Trace messages are described and listed in Appendix B, Trace Messages

Logging affects system performance. It is recommended that you use logging only as needed.

Debug messages should not be logged unless requested by the TIBCO Product Support Group. This option writes a lot of information to the log file and significantly reduces the speed of the adapter.

By default, the Use Advanced Logging checkbox is not checked. In this mode, you configure a standard log file using the fields in this tab.

If you check the **Use Advanced Logging** checkbox, you configure log sinks using icons in the TIBCO Designer project panel. This gives you complete control of selecting the destinations and associating the desired roles with each of the destinations.

Creating Log Sinks

To create log sinks, follow these steps:

1. Check the **Use Advanced Logging** checkbox.
2. Click **Apply**.
3. In the TIBCO Designer project panel, select the **Log Sinks** folder under the Advanced folder.
4. Select an existing log sink or create a new one:
   - Select the **fileSink** or **stdioSink** icon.
   - Create a new log sink by dragging and dropping the **GenericLog Sink** icon from the palette panel to the design panel. Then select a type in the Sink Type drop-down list in the Configuration tab. Click **Apply**.

5. With the desired log sink icon selected in the design panel, fill in the fields in the configuration panel. You can also change the name and enter a description for each sink by right-clicking on the sink icon in the project panel.
   - File sink logs the trace messages to a file. Specify the file limit, file count, and the option to append or overwrite. By default, the file limit is 30000 bytes, the file count is 3, and the mode is append.
   - STDIO sink sends trace messages to `stdout` or `stderr`. By default, `stdout` is selected.
   - Hawk sink sends each trace message to TIBCO Hawk Monitor or TIBCO Hawk Display using the Hawk session that is created by the adapter for monitoring purposes.
   - Network sink publishes each trace message on TIBCO Rendezvous. Specify the session and the subject on which the trace messages needs to be published.

Startup Tab

Do not change the setting in this tab. This tab displays the default startup behavior.

Table 8 Startup Tab

Field	Description
Show Startup Banner	The startup banner displays the runtime adapter version, the infrastructure version on which the adapter is built, and copyright information in the console window when the adapter is started.
Metadata Search URL	This field is predefined and cannot be changed. It specifies the location where the adapter searches for base schemas. All schemas that have been defined and saved at this location are loaded at startup.

Monitoring Tab

These settings do not need to be configured unless TIBCO Hawk is installed.

You can use microagents to supplement the monitoring information provided by the standard logging capability. Examples of supplemental information that you can obtain with microagents include the repository URL and the command line arguments used to start the adapter.

Click **Apply** before leaving this dialog to apply the changes.

See Chapter 7, Monitoring the Adapter Using TIBCO Hawk for a list of all supported microagents.

Many of the following fields can use global variables. Click the **Global Variables** tab in the project panel to add or modify a global variable.

Table 9 Monitoring Tab

Field	Description
Enable Standard Microagent	This field allows you to turn on or off the standard TIBCO Hawk Microagent. Clicking the globe icon changes the checkbox to a text field, allowing you to specify a global variable. When this is a text field, turn the microagent on and off by entering **`true`** or **`false`**.
Standard Microagent Name	This is the name for the standard microagent that will be registered with the TIBCO Hawk system. In most cases the default value is used, `COM.TIBCO.ADAPTER.adkenan.%%Deployment%%.%%InstanceId%%`. The value for the `%%deployment%%` and `%%InstanceId%%` global variable can be set or modified by selecting the session icon and then clicking the **Global Variables** tab in the project panel.
Standard Microagent Timeout	This field allows you to specify the amount of time the Hawk Agent should wait for HMA method invocations to complete before timing them out. The default is 10000 milliseconds. Normally there is no need to change this value. However, on machines under extreme stress where method invocations are timing out, this new option allows the timeout value to be increased.
Enable Class Microagent	This field allows you to configure how the class microagent is turned on and off. In most cases, the default setting `%%HawkEnabled%%` is used. Clicking the globe icon changes the text field to a checkbox. It allows you to turn the instance-specific or class-specific standard TIBCO Hawk Microagent on or off .
Class Microagent Name	This is the name for the class microagent that will be registered with the TIBCO Hawk system. In most cases the default value is used, `COM.TIBCO.adkenan.%%Deployment%%.%%InstanceId%%`.
Class Microagent Timeout	This field allows you to specify the amount of time the Hawk Agent should wait for HMA method invocations to complete before timing them out. The default is 10000 milliseconds. Normally there is no need to change this value. However, on machines under extreme stress where method invocations are timing out, this new option allows the timeout value to be increased.
Default Microagent Session	This field is predefined and cannot be changed. The session is automatically generated by TIBCO Designer and will be used by the standard, class, and custom microagents.

Adapter Services Options

TIBCO ActiveMatrix Adapter for Kenan/BP only provides request-respond service. Typically, an adapter instance can contain one or more request-respond services.

Request-Response Service Tabs

This service is often called a Request Reply Server or RPC (Remote Procedural Call) Server. When running as a Request-Response Service, the adapter receives requests from other TIBCO ActiveEnterprise applications and parses them. The output fields are wrapped in a schema and sent back to the caller.

You can configure parameters in the following tabs:

Configuration Tab

Specify the following options in the Configuration tab.

Table 10 Configuration Tab

Field	Description
Name	You can use the default name or replace it with a name of your choice.
Description	Optional. You can enter a description for the request-response service.
Transport Type	Select the transport type (JMS or TIBCO Rendezvous) to be used by the runtime adapter. This selection determines which options appear in the Transport tab. The transport can be configured to use a trusted store and identity resource for use in SSL (Secure Sockets Layer) configurations. TIBCO Rendezvous sessions and JMS topics have an SSL configuration field which uses a dialog to perform the SSL configuration. To enable and configure SSL, in the Project panel, expand the **Advanced** folder, then expand the **Sessions** folder. Select the **TIBCO Rendezvous** session or **JMS** topic and click **Use SSL?**. The SSL configuration options are explained in the online help associated with the session dialog. Click the question mark to display the online help.

Header Fields Tab

This tab allows you to specify the Header fields. While configuring the input XML message using an external application like TIBCO ActiveMatrix BusinessWorks, you need to configure only the API specific XML block.

Table 11 Header Fields Tab

Field	Description
Specify Header Fields	If checked you can specify the header fields in this tab. The runtime adapter will take the values that you have specified in this tab. If unchecked, the runtime adapter will use the default Kenan/BP header values. Note that you can also specify the header values in a request message or in a request response service activity when configuring a TIBCO ActiveMatrix BusinessWorks process. The runtime adapter can extract the header elements that you have specified at process or service level.
Account Server	This is an integer field. The value specified here denotes which customer server is expected to receive the incoming request.
Application Name	This is a string field. This contains details about the Application making the call to the Kenan/FX API set.
Operator Name	This is a string field. It is used in conjunction with the Application Name field.
Max Count	This is an Integer field. The field allows you to specify the maximum number of records.
Log Level	This is an Integer field. This field defines the amount of system information you log for an API TS request. The valid values are as follows: ---0 = none ---1 = low ---2 = medium ---3 = high
Call Correlation	This is a string field. This field allows you to specify an identifier for your client application. The value is written to logs. By setting it differently for different clients, you can keep track of transaction volume by client.
External TransactionXRef	This is a string field. This field allows you to specify an identifier for a transaction. This field is designed to help you identify transactions within the log file. You can use this field to assign a different value for each transaction (group of calls) that you have created.
Execution Trace	This is a Boolean field. It is used to denote whether the `<TraceResults>` element should be included in the response to the method call. It helps in tracing each transaction with the Kenan/FX middleware.

Table 11 Header Fields Tab

Field	Description
Debug Flag	This is a Boolean field. It lets you specify whether debug information should be written to log files. On a message basis, it overrides the debug settings in the `KenanFX.ini` file.
Block Size	This is an Integer field. It lets the user specify the maximum number of records for each query to return. On a message basis, it overrides the BlockSize setting in the `KenanFX.ini` file.

A Valid XML Message Example

An example of a valid XML message which can be successfully processed by Kenan/BP is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<Request>
   <Header>
      <AccountServer e-dtype="int">3</AccountServer>
      <ApplicationName e-dtype="string">Kenan</ApplicationName>
      <OperatorName e-dtype="string">Adapter</OperatorName>
      <ExecutionTrace e-dtype="boolean">false</ExecutionTrace>
      <DebugFlag e-dtype="boolean">false</DebugFlag>
      <BlockSize e-dtype="int">100</BlockSize>
      <MaxCount
e-dtype="int">oracle.xml.parser.v2.XMLElement@7433b121</MaxCount>
      <LogLevel
e-dtype="int">oracle.xml.parser.v2.XMLElement@6db22920</LogLevel>
      <CallCorrelation
e-dtype="string">oracle.xml.parser.v2.XMLElement@4baa2c23</CallCor
relation>
      <ExternalTransactionXref
e-dtype="sting">oracle.xml.parser.v2.XMLElement@1137d4a4</External
TransactionXref>
   </Header>
   <AccountGet xmlns="CSG">
    <Account>
        <Key>
            <AccountInternalId
e-dtype="int">339</AccountInternalId>
        </Key>
    </Account>
</AccountGet>
</Request>
```

As shown in the XML block, the entire XML request is enclosed in `<Request></Request>` tags. In this, there are two other XML blocks enclosed in `<Header></Header>` and the `<AccountGet></AccountGet>` nodes respectively.

The Header block contains static information about the nature of the call to the Kenan middleware. The second node is the one containing business entity (for example, Account, Product) and operation (Get, Create) specific information. The node and its contents will change depending on the API call being executed.

The adapter supports a feature wherein you can specify the fields under the `<Header></Header>` tags (also referred to as header fields) separately from the original XML request. By doing so, you have the option to configure only API specific fields as part of the incoming XML message. This can be done through the Header Fields tab.

For more information about the parameters in the Header Fields tab, refer to the Header Fields section of the Request Schema chapter in *API-TS Guide*.

Transport Tab

Message Transport options can be set for the Request-Response Service depending on the transport type selected in the Configuration tab of the request-response service.

The following options appear in the Transport tab if you select Rendezvous as the transport type in the Configuration tab:

Table 12 Transport Tab (Rendezvous)

Field	Description
Message Subject	By default, a service uses a message subject that is generated using the `Domain` and `Deployment` global variables, the adapter acronym, the adapter instance name, and the service name. If you use this default subject, make sure the values for `Domain` and `Deployment` are not empty. You can type a Rendezvous subject name different from the default in this field. See the TIBCO Rendezvous documentation for information about specifying subject names

Table 12 Transport Tab (Rendezvous)

Field	Description
Quality of Service	Select the level of service that determines how messages are sent. TIBCO Rendezvous supports the following quality of services ---Reliable Reliable message delivery ensures that each multicast or broadcast message is received as long as the physical network and packet recipients are working. It also ensures that the loss of a message is detected. This choice can compensate for brief network failures because it can retransmit a message on request if the first attempt failed. This choice is appropriate when message delivery is expected but some loss can be tolerated. Messages are received without explicit confirmation. ---Certified Certified message delivery offers stronger assurances of message receipt, along with tighter control, greater flexibility, and fine-grained reporting. Guarantees that every certified message reaches its intended recipient in the order sent. The message can be sent across network boundaries, and if a network fails, delivery attempts continue until delivery succeeds or until the message's time limit expires. If certified message delivery is used, data is stored in a ledger file. The size of the ledger depends on several factors, the most important of which is the retention rate of stored data. That means that the ledger grows fastest in response to the cumulative length of undeliverable messages. You must ensure that sufficient disk space is available for the expected size of the ledger. ---Distributed Queue A distributed queue is a group of cooperating transport objects, each in a separate process. To obtain load balancing among servers, the adapter uses distributed queues for *one-of-n* delivery of messages to a group of servers. Each member of a distributed queue listens for the same subject using the TIBCO Rendezvous `Distributed` Queue listener objects. Even though many members listen for each inbound message (or task), only one member processes the message. For details on distributed queues, see the TIBCO Rendezvous documentation.
Wire Format	The wire format in which data will be sent. If you select TIBCO Rendezvous as the transport type, only the ActiveEnterprise Message wire format is available. `ActiveEnterprise Message` is an externally-described XML wire format supported by the TIBCO Adapter SDK. Control information for validation is sent in the message. If no control information is included, an exception is returned to the subscriber. TIBCO ActiveEnterprise standard wire format provides class information and packing rules for the TIBCO Adapter SDK set of data types. This format allows TIBCO ActiveEnterprise components to perform extra validation on messages sent or received. XML allows you to retrieve data as XML documents and metadata as XML Schemas (XSD). See the *TIBCO Adapter SDK Programmer's Guide* for details about the control information generated and sent with TIBCO ActiveEnterprise messages. Note that services must use the same wire format to exchange data, otherwise an error will occur.

Table 12 Transport Tab (Rendezvous)

Field	Description
Session Reference	When you create a service, TIBCO Designer creates a corresponding session resource in the Advanced > Sessions folder and displays it in this field. If you have explicitly created a custom session of the same type, you can click the **Browse** button to replace the auto-created session.
Endpoint Reference	This field is an endpoint reference for the service. It is a disabled field and points to the corresponding endpoint resource in the Advanced > Sessions folder. The endpoint resource is automatically created by TIBCO Designer.

The following options appear in the Transport tab if you select JMS as the transport type in the Configuration tab.

Table 13 Transport Tab (JMS)

Field	Description
Destination	By default, a service uses a dynamic destination that is generated using the `Domain` and `Deployment` global variables, the adapter acronym, the adapter instance name, and the service name. If you use this default dynamic destination, make sure the values for `Domain` and `Deployment` are not empty. You can override the default dynamic destination by specifying the static destination in this field. The static destination must be defined on the EMS server before it can be used by the runtime adapter. See *TIBCO Enterprise Message Service User's Guide* for information on destinations.
Wire Format	The wire format in which data will be sent. If you select JMS as the transport type, only THE `XML Message` wire format is available. The `XML Message` wire format conforms to specifically constructed and fully compliant XML Schema (XSD) based on the existing definition of the TIBCO ActiveEnterprise schema.
Connection Factory Type	TIBCO Enterprise Message Service supports the following connection types: ---Queue Point-to-point messaging. A message sent to a queue is consumed by one and only one receiver. Each message has only one receiver though multiple receivers may connect to the queue. The first receiver to access the queue gets the message. The other receivers do not. ---Topic Publish-subscribe messaging. A message published to a topic is broadcast to one or more subscribers. All messages published to the topic are received by all services that have subscribed to the topic.

Table 13 Transport Tab (JMS)

Field	Description
Delivery Mode	This field appears only if you select Topic in the Connection Factory Type drop-down list. Delivery mode is the method of delivery for a JMS message. The semantics for these fields are somewhat more complex than the explanation given here. See the *TIBCO Enterprise Message Service User's Guide* for more information. ---Durable Durable indicates that the service is registered with the EMS server. Messages sent to a durable subscription service are held by the EMS server until they are consumed by the service. Even if the service is down, the messages will be received and processed once the service is up. ---Non Durable Non Durable indicates that the service is not registered with the EMS server. Messages sent to a non-durable subscription service are not held by the EMS server. If the service stops working, it will not receive the messages that arrived at the EMS server while the service was down. Messages sent with the durable delivery mode are always written to persistent storage, except when they are published to a topic that has no durable subscribers. When a topic has no durable subscribers, there are no subscribers that need messages to be re-sent in the event of a server failure. Therefore, messages do not need to be saved, and performance is improved because disk I/O is not required.
Session Reference	When you create a service, TIBCO Designer creates a corresponding session resource in the Advanced > Sessions folder and displays it in this field. If you have explicitly created a custom session of the same type, you can click the **Browse** button to replace the auto-created session.
Endpoint Reference	An endpoint reference for the service. This is a disabled field and points to the corresponding endpoint resource in the Advanced > Sessions folder. The endpoint resource is automatically created by TIBCO Designer.

Schema Tab

The Schema tab only has the Class Reference field.

Table 14 Schema Tab

Field	Description
Class Reference	The Class Reference is set automatically. This field is read-only

Chapter 4 Transaction Handling

This chapter explains the User Defined Transactions (UDT) and Orders Services functionalities.

Topics

Overview

TIBCO ActiveMatrix Adapter for Kenan/BP provides Transaction support through User Defined Transactions (UDT) and Orders Services.

Kenan/BP User-Defined Transactions (UDT) is a premium module of the Kenan FX suite. It supports grouping of multiple API requests into a single request. It is a convenient way to implement transactional integrity. If any request fails for some reason, the entire block of requests is rolled back. In addition, these provide a higher level of abstraction to the user from the Kenan/BP API-TS.

TIBCO ActiveMatrix Adapter for Kenan/BP provides GUI tools that help you construct UDT requests with TIBCO ActiveMatrix BusinessWorks XML utilities. The adapter also bundles new examples to demonstrate the usage of UDT. See the *TIBCO ActiveMatrix Adapter for Kenan/BP Examples* for more information.

Orders Services is a mandatory component of the Kenan FX middleware. It is fully supported by the adapter through the standard Kenan API Transaction Set. Orders Services adds transaction integrity and manages objects that are covered by the Ordering Module such as Account, Service, and NCRs.

See *Kenan User Defined Transactions Guide* for details about UDT. See *Kenan Orders Services Guide* for details about Orders Services.

UDT Functionality

User-defined transactions (UDTs) allow you to combine several requests into a single aggregate request. The API Transaction Set processes this aggregate request as a unit, providing transactional integrity at the overall request level.

TIBCO ActiveMatrix Adapter for Kenan/BP uses the Document Object Model (DOM) interface to execute standard API-TS and UDT requests.

A generic UDT service is available with the purchase of the UDT module. Direct parsing allows users to parse the UDT response data in a simple way.

Advantages of Using UDT

User-defined transactions offer the following benefits:

- They are fast. It is much faster to combine several requests into one and execute them all at once than to send requests individually, one after the other.
- They simplify the process of making requests. It is easier to keep track of related requests when they are grouped within a user-defined transaction. You can even design user-defined transactions that mirror the content of a GUI, so that all of the information on the screen at a given time is sent and processed together.
- They provide a higher level of abstraction than the more granular API objects.
- They maintain transactional integrity at the overall request level. If for some reason one part of a user-defined transaction fails, the entire transaction is rolled back.

UDT Service CustomerUdtRequest and AdminUdtRequest are shipped with the Kenan FX UDT module. You can also customize a UDT Service to perform different transactions, for example, to image a transaction process of a particular GUI screen.

UDT Example

An example snippet of a UDT call is as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Request xmlns:aruba="ARUBA" xmlns:csg="CSG"
xmlns:xsi="xmlSchema">
<Header>
    <AccountServer e-dtype="int">3</AccountServer>
    <ApplicationName e-dtype="string">null</ApplicationName>
    <OperatorName e-dtype="string">null</OperatorName>
</Header>
```

```
<CustomerUdtRequest>
<RequestList e-dtype="list">
    <Account>
        <AccountGet>
        <Account>
        <Fetch e-dtype="boolean">true</Fetch>
        <Key>
        <AccountInternalId e-dtype="int">87</AccountInternalId>
        </Key>
        </Account>
        </AccountGet>
        <RequestId e-dtype="string">RootAccount</RequestId>
    </Account>
    <Account>
    <AccountBalanceSummary>
    <Account>
    <Key>
    <AccountInternalId>
    <Ref
e-dtype="string">RootAccount/Account/Key/AccountInternalId</Ref>
    </AccountInternalId>
    </Key>
    </Account>
    </AccountBalanceSummary>
    <RequestId e-dtype="string">AccountBalanceSummary1</RequestId>
    </Account>
    <Nrc>
    <NrcFind>
    <Nrc>
        <Fetch e-dtype="boolean">true</Fetch>
        <BillingAccountInternalId>
            <Equal>
            <Ref
e-dtype="string">RootAccount/Account/Key/AccountInternalId</Ref>
            </Equal>
        </BillingAccountInternalId>
    </Nrc>
    </NrcFind>
    <RequestId e-dtype="string">NrcFind1</RequestId>
    </Nrc>
</RequestList>
</CustomerUdtRequest>
</Request>
```

In the above XML block, the following sequence of events take place:

1. The details for an account with `AccountInternalId = 87` is retrieved from the Kenan/BP database.
2. The `AccoutBalanaceSummary` is calculated for the same Account.
   Here what has to be noted is that the `AccountInternalId` is given as:
   `<Ref e-dtype="string">RootAccount/Account/Key/AccountInternalId</Ref>` And not as an absolute number.

Moreover, the initial XML block (ref pt. 1) has
`<RequestId e-dtype="string">RootAccount</RequestId>.`
This means all subsequent references to this account can be made with the RootAccount symbol.

3. Similarly, the next block finds out the Non-recurring charges for this account. Here too, we see that the `BillingAccountInternalId` has been equated to `RootAccount/Account/Key/AccountInternalId.`

This is an example of UDTs at work. This method of one request accessing variables from another request is also known as the X-Path method. There are several ways to refer to data from other requests of the same UDT call:

- XPath
- SubRequest
- InputRequest
- InputRequestList

Refer to the UDT documentation for more usage details.

Enabling User-Defined Transactions

The generic UDT Service provided by Kenan is called `CustomerUdtRequest` with a corresponding XSD Schema called `CustomerUdtRequest.xsd`; the `CustomerUdtRequest.xsd` file has references to over 60.xsd schema files, so the schema reference for UDT covers all available operations using UDT, and is a fairly large set of.xsd files.

A UDT customization tool is provided to reduce the size of the schema and modify the XSD elements for compatibility with TIBCO Designer XML utilities.

The tool pops up a dialog prompting the user to select objects they need to use in the project. Only the user selection is imported into the project.

The UDT process is majorly composed of two steps: UDT trimming and UDT validation.

The UDT Tool assumes that all XSD files keep the original CSG folder structure. That is because the UDT tool needs to load referred XSD files, and it uses CSG's folder structure to find the related files.

Using the UDT Tool

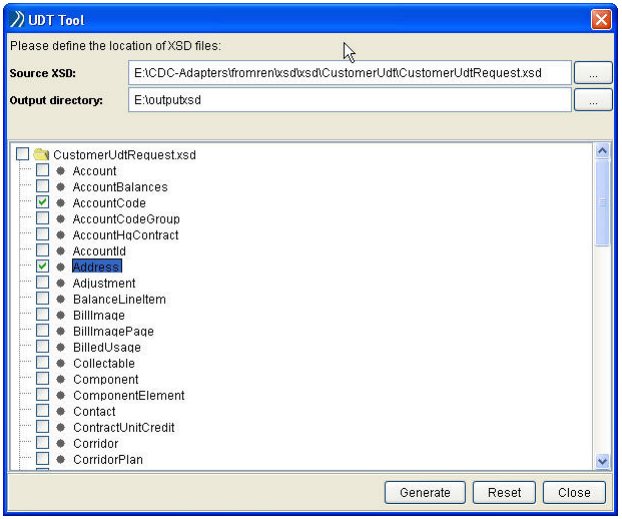
You can use the UDT tool from the TIBCO Designer. First set up a basic Kenan Adapter Configuration and follow these steps to use the UDT functionality.

1. From the UDT Tool menu in TIBCO Designer, select **Load UDT Tool**.
2. Define the XSD Settings.
   a. Browse and select the Source XSD file. Once you load the source XSD, the various elements are displayed.

The Source XSD file should be a `CustomerUdtRequest.xsd` or `AdminUdtRequest.xsd` provided by Kenan/BP and in the original Kenan/BP folder structure.

   b. Choose the elements you want to include in your custom XSD file (for example, Account, Contact, Invoice etc).
   c. Select the checkbox beside the element to include it in the XSD.
   d. Browse and select a location for the target XSD.
   e. Click the **Generate** button.

Figure 2 Generating the XSD Files



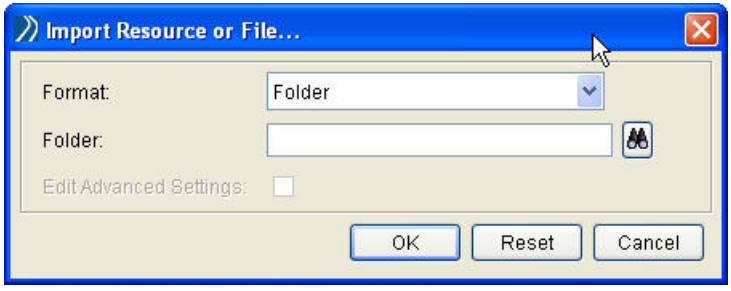
The custom XSD files will be created in the output folder specified. The following files will be generated to the output folder when the source file `CustomerUdtRequest.xsd` is used:

- `CustomerUdtRequest.xsd`
- `CustomerUdtResponse.xsd`
- `Request.xsd`
- A set of referenced XSDs according to the user selection.

A `Ref` XML element is added to `request.xsd`. This allows the user to construct UDT requests with XPath referencing.

3. Import the custom XSD files.
   a. Go back to TIBCO Designer and from the **Project** menu, select **Import Resources from File, Folder, URL..**
   b. In the Import Resource or File dialog box, select **Folder** from the Format drop down menu and browse to select the folder that contains the custom XSD files (the same folder that you specified as output directory when generating the custom XSD files). Click **OK**.

Figure 3 Import XSD Files



XPath and Element Reference

The XPath language provides a means of navigating to a particular element in an XML document. The portion of the XPath language used in UDTs lets you identify an XML element within a DOM using a path, much like a directory path in a file system.

Within a UDT, each request has a `<RequestId>` element that gives a unique `name` to that request. You can use this name (within an XPath path) to access elements from a previous request within that call.

In the Request List, when an element has two operations, and one operation depends on the other, you need a mechanism to create a reference.

To understand this, consider the following example:

1. Create a request that provides some data for subsequent requests to refer to. For example, use an `<AccountGet>` call and request the ID for an account (using the `<RequestID>` element).
2. Create a subsequent request within the same UDT call that refers to something in the object returned by the first request. For example, an `<AccountBalanceSummary>` call.

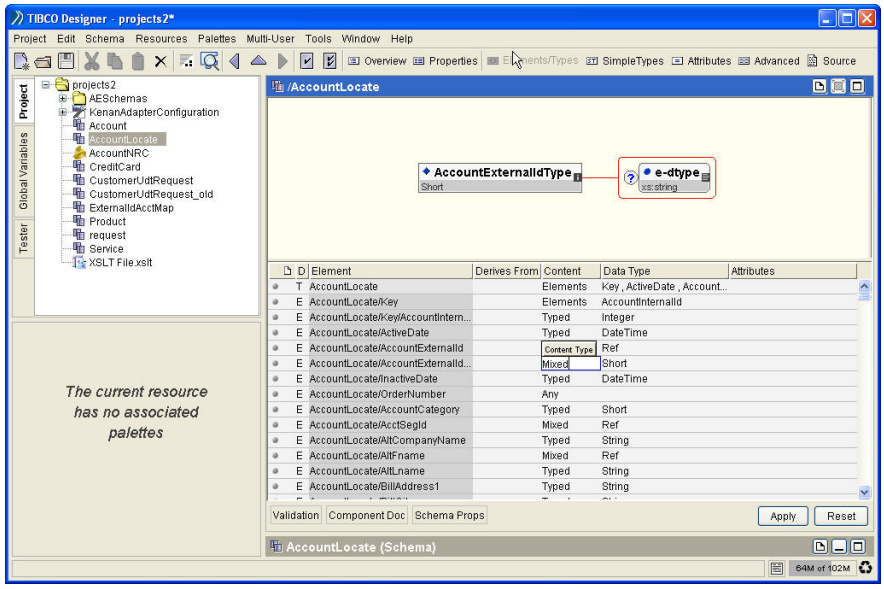
   In this case, the second request depends on data received from the first request. The `AccountBalanceSummary` Call retrieves the balance of the account, which was retrieved in the first request (`AccountGet`) in the UDT Call.

Steps for Element Reference

To do this, follow these steps:

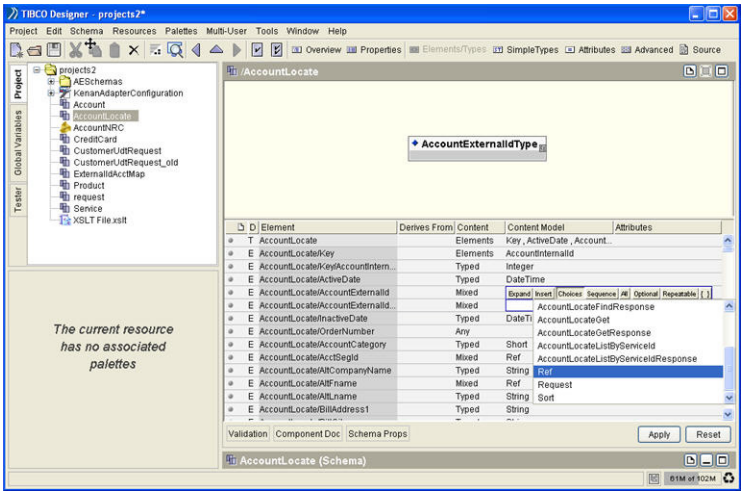
1. Open a Project in TIBCO Designer.
2. In the Project tab, click the XML Schema file and then click the **Overview** button in the menu.
3. In the right hand pane, navigate to the element for which you need to define a reference.
4. Click the corresponding content type for that element in the Content column. You will then see the Content type button. Click it and change the Content Type to Mixed.

Figure 4 Element Reference



5. You will then get another column Content Model. Click here and then click the **Insert** button and select **Ref**.

Figure 5 Insert the Reference



The `Ref` element is defined in the UDT tool-generated '`request.xsd`'. If the new xsd file set has been generated outside of the UDT tool and imported into Business Works, the original `request.xsd` will not include this `Ref` element, and `Ref` will not display.

6. Click **Apply** to save changes.

This element can now reference another element.

An Example of a Customized CustomerUDTRequest Schema

Here is an example of a customized `CustomerUDTRequest.xsd` file, it has fewer referred schema files and a simple structure:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<!--Generated by Turbo XML. Conforms to w3c
http://www.w3.org/2001/XMLSchema-->
<xsd:schema xmlns = "CSG"
 targetNamespace = "CSG"
 xmlns:xsd = "http://www.w3.org/2001/XMLSchema"
 elementFormDefault = "qualified"
 attributeFormDefault = "unqualified">
<xsd:include schemaLocation = "Account.xsd"/>
<xsd:include schemaLocation = "AccountBalances.xsd"/>
<xsd:include schemaLocation = "AccountHqContract.xsd"/>
<xsd:include schemaLocation = "AccountId.xsd"/>
<xsd:include schemaLocation = "BalanceLineItem.xsd"/>
<xsd:include schemaLocation = "Nrc.xsd"/>
<xsd:element name = "CustomerUdtRequest">
```

```
   <xsd:complexType>
   <xsd:sequence>
   <xsd:element name = "RequestList">
   <xsd:complexType>
 <xsd:sequence>
   <xsd:element name = "Account" type = "AccountRequest"/>
   <xsd:element name = "Nrc" type = "NrcRequest"/>
 </xsd:sequence>
<xsd:attribute name = "e-dtype" fixed = "list" type =
"xsd:string"/>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:complexType name = "RequestHeader">
<xsd:sequence>
<xsd:element name = "RequestId">
<xsd:complexType>
<xsd:simpleContent>
<xsd:extension base = "xsd:ID">
<xsd:attribute name = "e-dtype" fixed = "string" type =
"xsd:string"/>
</xsd:extension>
</xsd:simpleContent>
</xsd:complexType>
</xsd:element>
<xsd:element name = "InputRequest" minOccurs = "0">
<xsd:complexType>
<xsd:sequence minOccurs = "0">
<xsd:element name = "RequestId">
<xsd:complexType>
<xsd:simpleContent>
<xsd:extension base = "xsd:IDREF">
<xsd:attribute name = "e-dtype" fixed = "string" type =
"xsd:string"/>
</xsd:extension>
</xsd:simpleContent>
</xsd:complexType>
</xsd:element>
<xsd:element name = "RelationshipId" minOccurs = "0">
<xsd:complexType>
<xsd:simpleContent>
<xsd:extension base = "xsd:IDREF">
<xsd:attribute name = "e-dtype" fixed = "string" type =
"xsd:string"/>
</xsd:extension>
</xsd:simpleContent>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name = "AccountRequest">
<xsd:complexContent>
```

```
<xsd:extension base = "RequestHeader">
<xsd:sequence minOccurs = "0">
<xsd:element ref = "AccountGet"/>
<xsd:element ref = "AccountBalanceSummary"/>
<xsd:element name = "SubRequest" minOccurs = "0">
<xsd:complexType>
<xsd:sequence maxOccurs = "unbounded">
<xsd:sequence>
<xsd:element name = "RelationshipId" type = "String" fixed =
"Nrc_AccountBalances_Ref"/>
</xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
</xsd:element>
<xsd:element name = "NrcRequest" minOccurs = "0">
<xsd:complexType>
<xsd:complexContent>
<xsd:extension base = "NrcRequest">
<xsd:sequence>
<xsd:element name = "RelationshipId" type = "String" fixed =
"Nrc_Parent"/>
</xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
</xsd:schema>
```

An Example of Customized CustomerUDTResponse Schema

Here is an example of a customized `CustomerUDTResponse.xsd` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="CSG" targetNamespace="CSG" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xsd:include schemaLocation="Account.xsd" />
  <xsd:include schemaLocation="Product.xsd" />
  <xsd:include schemaLocation="request.xsd" />
  <xsd:include schemaLocation="AsyncRequest.xsd" />
  <xsd:include schemaLocation="BatchRequest.xsd" />
  <xsd:include schemaLocation="BulkOrderRequest.xsd" />
  <xsd:element name="UDTResponse">
    <xsd:complexType>
      <xsd:all>
        <xsd:element name="UDTResponseList">
```

```
            <xsd:complexType>
              <xsd:attribute name="list" />
              <xsd:choice>
                <xsd:element ref="AccountSequenceGetResponse" />
                <xsd:element ref="AccountCreateResponse" />
                <xsd:element ref="AccountGetResponse" />
                <xsd:element ref="AccountExtendedDataFindResponse" />
                <xsd:element ref="AccountChildCountResponse" />
                <xsd:element ref="AccountFindResponse" />
                <xsd:element ref="AccountFindByOrderNumberResponse" />
                <xsd:element ref="AccountFindCountResponse" />
                <xsd:element ref="AccountUpdateResponse" />
                <xsd:element ref="AccountActivateResponse" />
                <xsd:element ref="AccountReactivateResponse" />
                <xsd:element ref="AccountBalanceSummaryResponse" />
                <xsd:element ref="NetworkStatusResponse" />
                <xsd:element ref="AccountActiveChildCountResponse" />
                <xsd:element
ref="AccountFindWithExtendedDataResponse" />
                <xsd:element ref="ProductCreateResponse" />
                <xsd:element ref="ProductGetResponse" />
                <xsd:element ref="ProductAlternateGetResponse" />
                <xsd:element ref="ProductExtendedDataFindResponse" />
                <xsd:element ref="ProductExternalFindResponse" />
                <xsd:element ref="ProductFindResponse" />
                <xsd:element ref="ProductUpdateResponse" />
                <xsd:element ref="CurrencyDeriveResponse" />
                <xsd:element
ref="ProductFindWithExtendedDataResponse" />
                <xsd:element ref="ProductDisconnectValidateResponse"
/>
                <xsd:element ref="ProductUpdateValidateResponse" />
                <xsd:element ref="ProductFindByServiceOrderResponse"
/>
                <xsd:element ref="ProductBillToReevaluateResponse" />
                <xsd:element ref="AsyncRequestCreateResponse" />
                <xsd:element ref="BatchRequestCancelResponse" />
                <xsd:element ref="BatchRequestCancelRunResponse" />
                <xsd:element ref="BatchRequestCloseResponse" />
                <xsd:element ref="BatchRequestCreateResponse" />
                <xsd:element ref="BatchRequestDeleteResponse" />
                <xsd:element
ref="BatchRequestParticipantDispositionSummaryResponse" />
                <xsd:element ref="BatchRequestFindResponse" />
                <xsd:element ref="BatchRequestGetResponse" />
                <xsd:element ref="BatchRequestReopenResponse" />
                <xsd:element ref="BatchRequestRestartResponse" />
                <xsd:element ref="BatchRequestResumeResponse" />
                <xsd:element ref="BatchRequestResumeAllResponse" />
                <xsd:element ref="BatchRequestScheduleResponse" />
                <xsd:element ref="BatchRequestSequenceGetResponse" />
                <xsd:element ref="BatchRequestSuspendResponse" />
                <xsd:element ref="BatchRequestSuspendAllResponse" />
                <xsd:element ref="BatchRequestTerminateResponse" />
                <xsd:element ref="BatchRequestUnscheduleResponse" />
                <xsd:element ref="BatchRequestUpdateResponse" />
                <xsd:element ref="BatchRequestWorkSubmitResponse" />
```

```
              <xsd:element
ref="BulkOrderRequestByAccountCreateResponse" />
              <xsd:element
ref="BulkOrderRequestByAccountGetResponse" />
              <xsd:element
ref="BulkOrderRequestByAccountUpdateResponse" />
              <xsd:element
ref="BulkOrderRequestByServiceCreateResponse" />
              <xsd:element
ref="BulkOrderRequestByServicetGetResponse" />
              <xsd:element
ref="BulkOrderRequestByServiceUpdateResponse" />
            </xsd:choice>
          </xsd:complexType>
        </xsd:element>
      </xsd:all>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Direct Parsing of UDT Response

The following is an example of the original UDT response data from Kenan/BP.

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<Request xmlns="CSG"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <Header>
      <AccountServer e-dtype="int">3</AccountServer>
      <CallCorrelation
e-dtype="string">oracle.xml.parser.v2.XMLElement@38717323</CallCor
relation>
   </Header>
   <UDTResponse>
      <UDTResponseList e-dtype="list">
           <UDTResponse>
            <Account>
               <AccountCategory e-dtype="int">2</AccountCategory>
                .....
               <VipCode e-dtype="int">0</VipCode>
            </Account>
            <RequestId e-dtype="string">AccountGet</RequestId>
            <RequestObjName
e-dtype="string">Account</RequestObjName>
           </UDTResponse>
           <UDTResponse>
            <Product>
               <ArchFlag e-dtype="boolean">false</ArchFlag>
               <AutoActivation e-dtype="int">0</AutoActivation>
                .....
              <ViewStatus e-dtype="int">1</ViewStatus>
            </Product>
            <RequestId e-dtype="string">ProductGet</RequestId>
```

```
            <RequestObjName
e-dtype="string">Product</RequestObjName>
            </UDTResponse>
      </UDTResponseList>
   </UDTResponse>
</Request>
```

The response result information for each request method is contained in a pair of XML elements: `UDTResponse`, which are marked in bold.

When the direct parsing function is enabled in TIBCO Designer, the adapter makes some modification to the structure of the original response data to simplify the data parsing process. The following is an example of response data with the direct parsing enabled. The UDT request is exactly the same with the one associated with the above mentioned example.

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<Request xmlns="CSG"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <Header>
      <AccountServer e-dtype="int">3</AccountServer>
      <CallCorrelation
e-dtype="string">oracle.xml.parser.v2.XMLElement@58c16b18</CallCor
relation>
   </Header>
   <UDTResponse>
      <UDTResponseList e-dtype="list">
         <AccountGetResponse>
            <Account>
               <AccountCategory e-dtype="int">2</AccountCategory>
               <AccountExternalId
e-dtype="string">339</AccountExternalId>
                .....
               <VipCode e-dtype="int">0</VipCode>
            </Account>
            <RequestId e-dtype="string">AccountGet</RequestId>
            <RequestObjName
e-dtype="string">Account</RequestObjName>
         </AccountGetResponse>
         <ProductGetResponse>
            <Product>
               <ArchFlag e-dtype="boolean">false</ArchFlag>
               .....
               <ViewStatus e-dtype="int">1</ViewStatus>
            </Product>
            <RequestId e-dtype="string">ProductGet</RequestId>
            <RequestObjName
e-dtype="string">Product</RequestObjName>
         </ProductGetResponse>
      </UDTResponseList>
   </UDTResponse>
</Request>
```

The response result information for each request method is contained in a pair of XML elements: `ProductGetResponse` or `AccountGetResponse`, which are marked in bold. You can use the response schema (`CustomerUdtResponse.xsd`) generated by the UDT Tools to parse this response easily using TIBCO ActiveMatrix BusinessWorks. Please refer to *TIBCO Adapter for Kenan/BP Example Guide* for more information.

Orders Services

Orders Services is a standard module and technical component delivered with Kenan/BP. Orders Services delivers functions that create and manage customer and service orders, track order status, trigger order fulfillment through Request Handling and Tracking, manage order revisions and cancellations, and activate customers at the appropriate time in Kenan/BP.

Orders Services creates and processes orders. The central concept in orders services is the order. An order is a logical grouping of one or more *service orders*. A service order is a grouping of one or more order items that are associated to a single service or account.

Service orders also enable partial order completion, which allows subparts of a large order to be activated in Kenan/BP without dependency on the fulfillment status of other parts of the order or the overall status of the order.

Order services are capable of providing certain levels of transactional support. Refer to the *Kenan Orders Services guide* for more information.

Orders Services Example

An orders services(`Order_Service_RV.zip`) example is shipped with TIBCO ActiveMatrix Adapter for Kenan/BP. You can find it in the *TIBCO_HOME*/adapter/adkenan/*version_num*/examples/BusinessWorks directory. Refer to *TIBCO Adapter for Kenan/BP Examples* chapter 7 for more information.

Chapter 5 Advanced Features

This chapter explains how to configure an adapter with advanced options.

Topics

Using Global Variables

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

Global variable substitution allows you to accomplish the following.

- Substitute string variables specified in the project at startup time.
- Locally define the value for a global variable for a specific project. The local value takes precedence over any global value.
- Specify the value for a variable in a properties file. This overrides the project repository and values set in code but not variables set on the command line.
- Enforce the predefined variables listed in Configuring a Remote Adapter on page 66.

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

Specifying Global Variables

The adapter can specify variables:

- In the project during configuration using TIBCO Designer.
- In a properties file.
- In TIBCO Administrator Enterprise Edition when deploying the project.

The values in the properties file or Enterprise Edition take precedence over the values set in the project through TIBCO Designer.

Specifying Variables Using TIBCO Designer

Global variables provide an easy way to set defaults for use throughout your project. There are several ways in which they can be used:

- Define a variable using TIBCO Designer, then override the value for individual applications at deployment time using TIBCO Administrator. You can also override values for predefined variables, unless the GUI does not allow you to set them later.
- Predefine a variable using TIBCO Designer, then override the value for individual services (for example, a publication service or a TIBCO ActiveMatrix BusinessWorks process) at deployment time using TIBCO

Administrator. The values you specify are then used at runtime. You can also override values for predefined variables, unless the GUI does not allow you to set them later.

For example, you could assign the value `7474` to the predefined global variable `RvDaemon`. You can then use the variable in different sessions in your adapter. If you want to change the TIBCO Rendezvous daemon for your adapter, you can globally set it to a different value or override it from the TRA file.

To Specify Global Variables:

1. In the project panel, select the **Global Variables** tab.

   The project panel is updated to display all currently defined global variables. Click **Open Advanced Editor** (pencil icon at the top left corner). You now have these choices:

   — To assign or change a variable value, select that region and triple-click the variable. The variable expands so you can change either the variable name or the variable value. Press Enter when you're done.

   — To add a new global variable group, click the group icon (on the left below the project panel). Specify the name of the group, then press Enter.

   — To add a global variable to a group, select the desired group icon and click the **abc** icon below the project panel.

   — To add a global variable to the list, click the **abc** icon below the project panel. A new global variable item is added to the bottom of the list. Type the variable name and, optionally, the value. Press Enter when done.

   The global variable is now displayed in the global variables list.

2. When you want to use the global variable in the fields of a resource, enter the variable name surrounded by `%%` on both sides.

   When the project is deployed and the configured components are run, all occurrences of the global variable name are replaced with the global variable value (unless it was overridden in a way that had higher precedence).

A number of global variables are predefined. See Configuring a Remote Adapter for details. You may add definitions of any new variables you need to the existing list.

Changing Global Variable Values at Runtime

You can change the value of a global variable when you deploy your project in TIBCO Administrator. See the TIBCO Administrator documentation on modifying runtime variables.

You can also specify values for global variables when starting a process engine from the command line. To do this, specify the following as a command line argument when starting the process engine:

`-tibco.clientVar.`*variablePathAndName value*

where *variablePathAndName* is the name of the variable you want to set, including the path to the variable if it is contained in a folder and *value* is the value you want the variable to use. For example, if you have a global variable named `item1` contained in a folder named `myGroup` and you want to set its value to 500, add the following argument to the command line when starting the process engine:

```
-tibco.clientVar.myGroup/item1 500
```

Predefined Global Variables

The next table lists and explains the predefined global variables. Some global variables are automatically used within the system when an adapter instance is configured.

Table 15 Predefined Global Variables

Variable	Description
`Deployment`	Defaults to the TIBCO Designer project name. This global variable is used by TIBCO Designer to partially define the subject name for a service.
`JmsProviderUrl`	Specifies where the EMS server is located. Setting this value mostly makes sense in the early stages of a project, when only one EMS server is used.
`DirLedger`	Specifies the path name of the TIBCO Rendezvous certified messaging ledger file. The default is the root installation directory.
`DirTrace`	Specifies the path name for the log file used by the adapter. The default is the root installation directory.
`Domain`	The default value for file-based local projects is `MyDomain`. The value for server-based projects is the domain to which the project was saved.
`HawkEnabled`	Indicates whether TIBCO Hawk is used to monitor the adapter. True indicates that a TIBCO Hawk microagent is defined for the adapter. False indicates the microagent is not to be used.
`JmsSslProviderUrl`	Specifies where the JMS SSL daemon is located.

Table 15 Predefined Global Variables

Variable	Description
`RemoteRvDaemon`	TIBCO Rendezvous routing daemon (`rvrd`) to be used. See *TIBCO Administrator Server Configuration Guide* for details about setting up a domain using `rvrd`.
`RvDaemon`	TIBCO Rendezvous daemon. Sessions use this daemon to establish communication. The default value is `7500`.
`RvNetwork`	TIBCO Rendezvous network. This variable need only be set on computers with more than one network interface. If specified, the TIBCO Rendezvous daemon uses that network for all outbound messages. In most cases, you can leave the default.
`RvService`	TIBCO Rendezvous service. The Rendezvous daemon divides the network into logical partitions. Each transport communicates on a single service. A transport can communicate only on the same service with other transports. Unless you are using a non-default TIBCO Rendezvous configuration, you should leave the default (7500).
`RvaHost`	Computer on which the TIBCO Rendezvous agent runs. This variable is only relevant if you are using the TIBCO Rendezvous Agent (`rva`) instead of the TIBCO Rendezvous daemon, and if you have configured a non-default setup. See *TIBCO Rendezvous Administration* for details about specifying rva parameters.
`RvaPort`	TCP port where the TIBCO Rendezvous agent (rva) listens for client connection requests. Defaults to 7501. See *TIBCO Rendezvous Administration* for details about specifying the rva parameters.
`TIBHawkDaemon`	TIBCO Rendezvous daemon used in the TIBCO Hawk session. See the *TIBCO Hawk Installation and Configuration* manual for details on this parameter.
`TIBHawkNetwork`	TIBCO Rendezvous network used by the TIBCO Hawk session. See the *TIBCO Hawk Installation and Configuration* manual for details on this parameter.
`TIBHawkService`	TIBCO Rendezvous service used by the TIBCO Hawk session. See the *TIBCO Hawk Installation and Configuration* manual for details on this parameter.

Configuring a Remote Adapter

TIBCO ActiveMatrix Adapter for Kenan/BP can be configured remotely in a scenario where TIBCO Designer is installed on your machine and the remote host has an installation of the adapter and the Kenan/BP Client. To do this:

1. In a command prompt window, copy the TIBCO ActiveMatrix Adapter for Kenan/BP palette files from the remote machine (located in the *TIBCO_HOME*/adapter/adkenan/*version_num*/lib directory).
   a. Go to the *TIBCO_HOME*/Designer/*version_num*/bin directory and open the designer.tra file. Modify the path specified for the property that enables TIBCO Designer to find the adapter's palette as follows:

      ```
      #application.args -d

      java.property.palettePath

      TIBCO_HOME/adapter/adkenan/version_num/lib/palettes
      ```

   b. Save and close the designer.tra file.
   c. Open the project in TIBCO Designer. In the project tree panel, drag and drop the **Kenan/BPAdapterConfiguration** icon from the palette panel to the design panel.

Using the Adapter with a Revision Control System

TIBCO Designer supports revision control systems such as Microsoft Visual SourceSafe and Perforce. If you are using a revision control system, you must manually add some configured resources to the revision control system and check in the resources when completing the instance configuration.

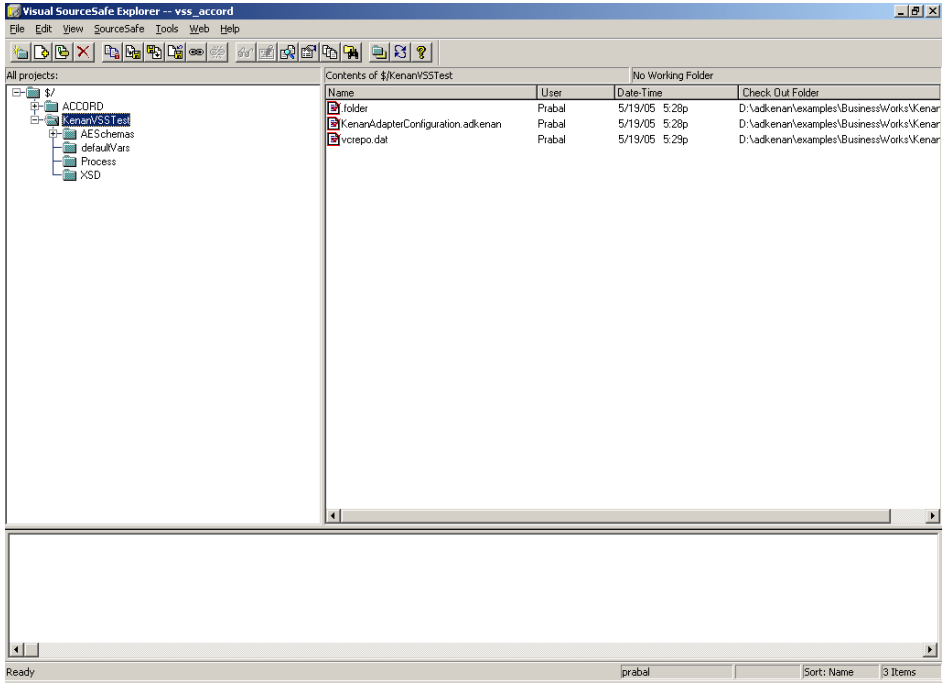
As part of service configuration, the adapter creates schema files in `root/AESchemas/ae/adkenan`. For example, if you configure a service in a `Kenan/BPAdapterConfiguration`, the following files are created:

```
Project_root /AESchemas/ae/adkenan/tdschmea.aeschema
Project_root /AESchemas/ae/adkenan/tdschmea(folder)
Project_root
/AESchemas/ae/adkenan/'Kenan/BPAdapterConfiguration.aeschema
Project_root'
/AESchemas/ae/adkenan/'Kenan/BPAdapterConfiguration(folder)'
```

`where Kenan/BPAdapterConfiguration is the adapter configuration.`

`The following images show the mentioned folders in a Revision Control System:`

Figure 6 The Kenan/BP Schema in a Version Control System



When the project is saved and a revision control system has been specified, the adapter displays a warning that additional files were created and should be added to the revision control system. This warning appears only when the files are created for the first time. The warning displays a Go To Resource button that helps in navigating to the resource. You should use the Multi-User>Add Resources to RCS menu command to add these files to the revision control system.

For information about on to use the Multi-User feature in TIBCO Designer, refer to the *TIBCO Designer User's Guide*.

Copy, Cut, Paste and Move Operations

To successfully copy and paste a service from adapter *Instance1* to *Instance2*, the adapter configuration and schema files for *Instance2* must be checked out.

To successfully cut and paste a service from adapter *Instance1* to *Instance2*, the adapter configuration and schema files for both *Instance1* and *Instance2* must be checked out.

To successfully move a service from adapter *Instance1* to *Instance2*, the adapter configuration and schema files for both *Instance1* and *Instance2* must be checked out.

Handling DATE Fields

You can handle DATE fields in TIBCO ActiveMatrix BusinessWorks. By default, TIBCO ActiveMatrix BusinessWorks accepts data for `dateTime` fields in the format `YYYY-mm-ddThh-mm-ss`. This format is not acceptable to the Kenan API-TS which expects the dateTime fields to be in the `YYYY-mm-dd hh-mm-ss` format. (Please note the space in between).

To use a custom Java code to transform the date fields:

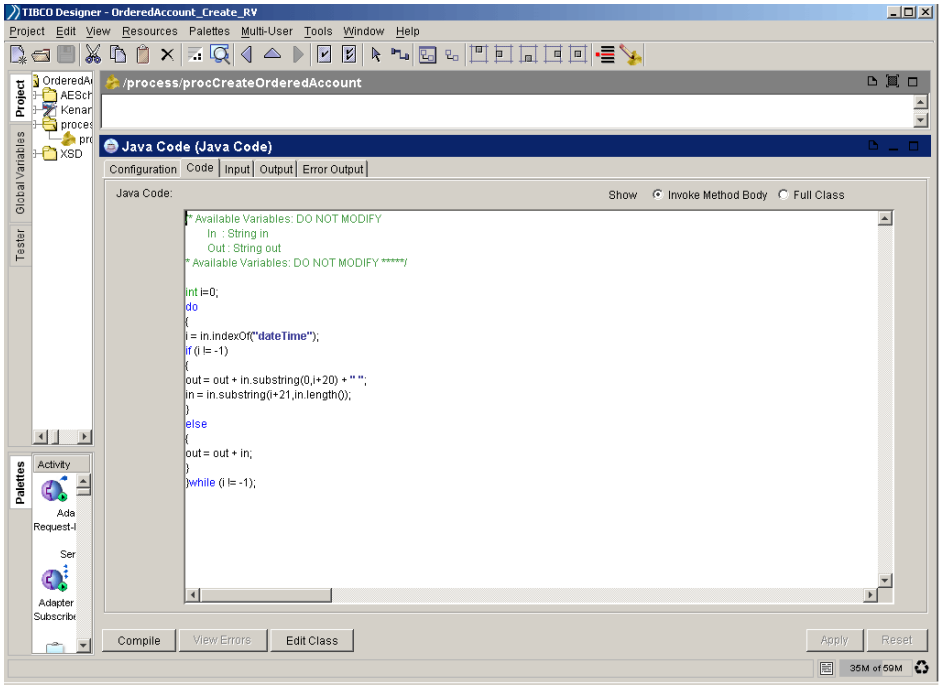
1. Drag and drop the **Java Code** activity from the palette panel to the design panel.
2. In the Configuration tab, click  to add an input parameter.
3. Type **`in`** in the Field name field. The **`in`** parameter means case sensitive.
4. Select **Required** in the Occurrence drop-down list.
5. Select **string** in the Type drop-down list.
6. Similarly, add an output parameter with the Field name as **`out`** (case sensitive), Type as **`string`**, and Occurrence as **Required**.
7. Select the **Code** tab. Select the **Invoke Method Body** radio button.
8. Copy and paste the following piece of code into the Java Code block.

```
int i=0;
    do
    {
        i = in.indexOf("dateTime");
        if (i != -1)
        {
            out = out + in.substring(0,i+20) + " ";
            in  = in.substring(i+21,in.length());
          }
          else
        {
            out = out + in;
        }
    }while (i != -1);
```

9. Click **Apply**.
10. Click **Compile**. You will see a dialog box saying `Code compiled successfully`.
11. Attach a trigger from the Render XML task to the Java Code task. Attach a trigger from the Java Code task to the Invoke Request-Response Service task.

12. Map the `XMLSTRING` output of the Render XML task to the `in` input of the Java Code.
13. Similarly, map the `out` output of the Java Code to the `InXMLMsg` input of the Invoke task.

Figure 7 Java Code



Custom Function Callout

TIBCO ActiveMatrix Adapter for Kenan/BP allows you to invoke custom functions.

Integrating the Custom Function with the Adapter

The adapter exposes some interfaces which can be extended to generate the required custom function. The name and the prototype of the custom function are decided beforehand by the adapter. The content of the function however is user specific.

When a request comes to the adapter, it validates whether it is a custom request. If it is a custom request, the name of the class which contains the custom function is extracted from the incoming request. The adapter then invokes the custom function which is part of the class specified in the incoming request.

Using Custom Function Callout

To use Custom Function Callout, follow these steps:

1. Configure a custom request XSD (XML Schema Definition). This can customize this schema as required. However two essential prerequisites for the same are:
   - The root element for the XSD would be `CustomRequest`. In the request XSD provided with the Kenan API installation, the same element is `Request`. All

the XSD sequences and the complex application types should be contained in a tag `<xs:element name = "CustomRequest">`.

— There should be an XSD block containing an attribute `CustomFunctionName`. The xsd should contain a block like this:

```
<xs:element name = "CustomFunctionName">

<xs:complexType>

<xs:simpleContent>

<xs:extension base = "xs:string">

<xs:attribute name = "e-dtype" fixed = "string" type =
"xs:string"/>

  </xs:extension>

  </xs:simpleContent>

  </xs:complexType>

  </xs:element>
```

2. Write the source code for the custom class.

   You can extend the interface provided by the adapter. TheTIBCO ActiveMatrix Adapter for Kenan/BP exposes two interfaces for executing custom function callout, which are `KenanMessageHandler` and `KenanMessageHandlerFactory`. There are two methods for consideration, both of which are exposed by the adapter:

   — `processMessage()` - This method incorporates custom function logic and is completely dependent on user requirements. This method is found in the `KenanMessageHandler` interface. The method accepts a DOM document and an XMLConnection object as input and returns a DOM document to the adapter.

   — `getMessageHandler()` - This method creates an object for handling the custom function callout operation. This method is found in the `KenanMessageHandlerFactory` interface. You have the flexibility to create a message handler either on every message or on every thread based on the incoming custom function handler name. This function returns a `KenanMessageHandler` class object. This method needs to be synchronized to make it thread safe.

3. Compile the source code that is used to generate the custom class.

   Compile the source code for the custom class. Once the source code for the custom function has been generated, it has to be compiled to generate the custom class. A sample compilation script is provided with the adapter installation in

`$TIBCO_HOME/adapter/adkenan/version_num/examples/KenanCustomCallout/CustomClasses.`

The compiled class has to be placed in the classpath so that the adapter can find it while running. The location of the custom classes is already a part of the `CUSTOM_CP_EXT` path in the adapter properties file.

4. Specify the name of the factory class in the adapter property file. The `adkenan.KenanCustomCallOutFactoryName` property should be uncommented and the Custom Factory class name should be provided. For example, set KenanCustomFactory to the `adkenan.KenanCustomCallOutFactoryName` property.
5. Use an external application like TIBCO ActiveMatrix BusinessWorks to configure a custom request for the adapter after the related adapter configuration is done.

   To do this, parse the custom request schema as explained in step 1. This is similar to the configuration of a single API request.

   Once the custom request reaches the adapter, it forwards the request data to the custom class. The custom class returns a DOM document as the response to the request. On receiving the response, the adapter serializes it and sends it as a reply.

Refer to the Custom Callout Example in *TIBCO ActiveMatrix Adapter for Kenan/BP Examples* for more information.

Using TIBCO ActiveMatrix BusinessWorks

This section includes procedures and prerequisites that have to be carried out to integrate the adapter with TIBCO ActiveMatrix BusinessWorks.

Configuring a TIBCO ActiveMatrix BusinessWorks Example

To configure a TIBCO ActiveMatrix BusinessWorks example:

Task A Create and Configure an adapter instance

1. Start TIBCO Designer and select **New Empty Project**.
2. In the Save Project dialog, click the **Browse** button to select the location of the project and click **OK**. A project is created.
3. Drag and drop the **KenanAdapterConfiguration** icon from the palettes panel into the design panel. An adapter instance is created and the default instance name is `KenanAdapterConfiguration`. You can change the instance name.
4. Click the **Global Variables** tab in the project panel and provide values for the following global variables:
   - `adkenanLogin` (The username used to connect to the Security Server)
   - `adkenanPassword` (The password used to connect to the Security Server)
   - `adkenanSecurityRealm` (The Security Realm variable for the Security Server)
5. Drag and drop the **Request-Response Service** icon from the palette panel into the design panel.
6. Save the project by selecting **Project > Save**.
7. Exit TIBCO Designer.

Task B Import the XSD files

1. Open the project saved in Task A.
2. Create a folder named `XSD` under the project root folder where you will store all imported XSD files.
3. Create another folder called `Process` to store the BusinessWorks process.
4. Import the following XSD files in the order specified: `request.xsd`, `AccountLocate.xsd`, `ExternalIdAcctMap.xsd`, `CreditCard.xsd`, and `Account.xsd`. The order is useful because `Account.xsd` includes the schema

location of the other XSD files. These XSD files are a part of the Kenan/BP installation.

5. To import an XSD file, select **Project > Import Resources from File, Folder, URL**. Select the file (.xsd,.xslt,.wsdl) from the drop-down list and browse to location of the XSD file.

Task C Configuring the BusinessWorks process

1. Create a new TIBCO ActiveMatrix BusinessWorks process by dragging and dropping the **Process Definition** icon from the palette panel to the design panel.
2. Configure a mapper task:
   a. Drag and drop a mapper task from the palette panel to the design panel.
   b. Click the **Input Editor** tab.
   c. Click the **Add Child** button and select **XML Element Reference** from the Content drop-down list.
   d. In the Schema field (which shows a default of `<No Namespace>`), browse and select the `AccountGet` element in `Account.xsd` as shown in Figure 8.
   e. Click **OK** in the Select a Resource dialog box.
   f. Click **Apply** in the Mapper task dialog box.
   g. Select the **Input** tab to provide the field values to execute this request.
   h. Expand the **AccountGet** node in the Activity Input panel and specify a value for the AccountInternal Id and specify **`int`** in the e-dtype field as shown in Figure 9.
   i. Click **Apply**.
   j. Attach a trigger from the Start node to the Mapper task.

Figure 8 Select a Resource

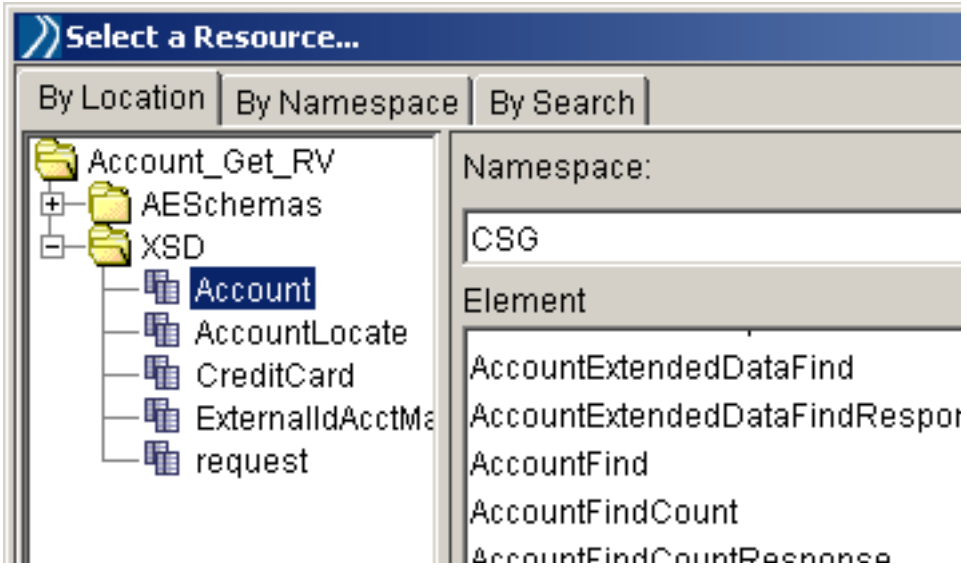
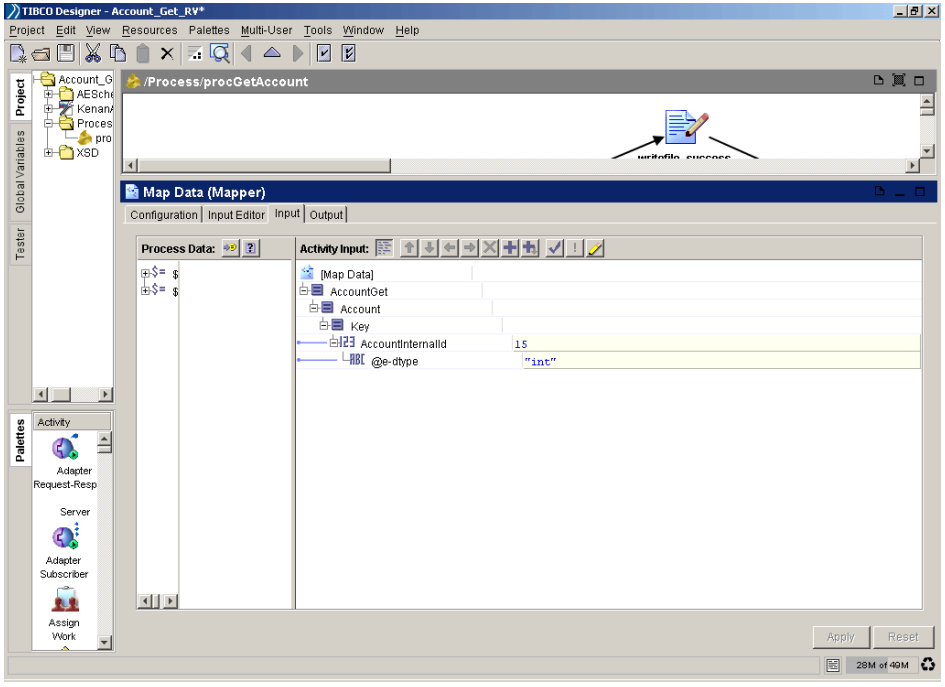
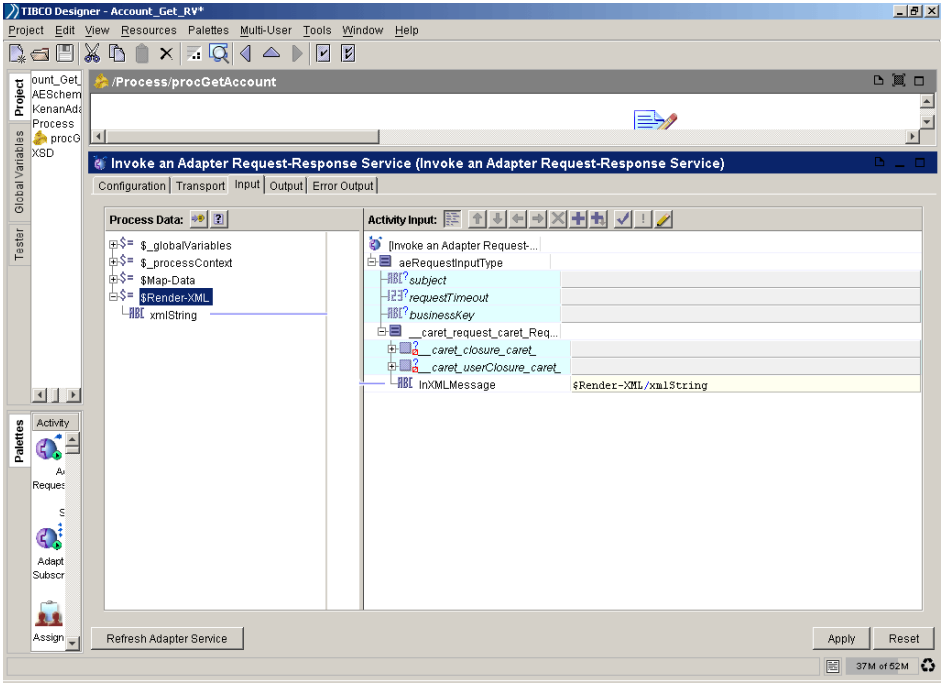


Figure 9 Map Data



3. Configure a Render-XML task
   a. Drag and drop a Render-XML task from the palette panel to the design panel.
   b. Attach a trigger from the Mapper task to the Render XML task.
   c. In the Configuration tab, select **text** in the Output Style drop-down list. Check the **Validate Input** and **Format with Default Namespace Prefix** checkboxes. Select **UTF-8** for encoding.
   d. In the Input Editor tab, select the Request element from `Account.xsd`. The procedure is exactly the one used in the Mapper task. (Refer to **step c** and step d in step 2).
   e. In the Input tab, provide the field values to execute this request.
   f. Expand the **Request** node in the Activity Input panel. Specify values for the fields under the Header group.
   g. Drag and drop the **AccountGet** node from the Process Data panel (this is the output of the mapper task) into the node named any element or exception.
   h. In the dialog box that follows, titled Mapping Wizard, select (any element). Click **Next** and then click **Finish**.
4. Configure an Invoke an Adapter Request-Response Service task
   a. Drag and drop the **Invoke an Adapter Request-Response Service** icon from the palette panel to the design panel.
   b. In the Adapter Service field, browse and select the Request-Response Service. Click **Apply**.
   c. Attach a trigger from the Render XML task to the Invoke task.
   d. Map the Render XML output (a string called `XMLstring`) to the incoming Request-Response Schemas (a string named `InXMLMsg`) as shown in Figure 10.
   e. Click **Apply**.

Figure 10 Invoke an Adapter Request-Response Service



Chapter 6 Deploying the Adapter Using TIBCO Administrator

This chapter provides an overview on deploying, starting, stopping, and monitoring adapters using the TIBCO Administrator web interface.

Topics

Creating an EAR File in TIBCO Designer

Generate an Enterprise Archive file (EAR) that contains information about the adapter services to deploy.

The EAR file contains information on what you want to deploy. This could be one or more adapter services, one or more TIBCO ActiveMatrix BusinessWorks process engines, or both.

Building an archive creates the EAR file, which you can then deploy from TIBCO Administrator. If you make changes to the business processes or adapter services included in the archive, you need to rebuild the archive. Saving the project does not affect the archive.

In TIBCO Designer, follow these steps to create an EAR:

1. Configure the adapter services.
2. Drag and drop the **Enterprise Archive** resource from the palette panel to the design panel.

   If there are any configured adapter services in your project, an Adapter Archive resource becomes available in the palette panel.
3. Drag the **Adapter Archive** into the design panel and specify information in the Configuration tab.
4. Click **Apply**.
5. Go to the Enterprise Archive and click **Build Archive** to create the archive file.

See Also

See the *TIBCO Designer User's Guide* for more information about this procedure. The guide is available from the Designer `Help` menu.

Deploying the Project

Before deploying a project, the machine on which the adapter is installed must be part of a TIBCO administration domain. After you have installed the TIBCO Administration Server, any machine on which you install TIBCO Runtime Agent (required by an adapter) can be added to the administration domain. The TIBCO software installed on the machine is then visible and accessible via the TIBCO Administrator GUI.

When you deploy a project, startup scripts and other information about the different components are sent to the machines to which the components were assigned. The project data store and TIBCO Administration Server are updated with the deployed components.

To deploy a project:

1. Import the EAR file into TIBCO Administrator Enterprise Edition.
2. Assign adapter archives in the EAR file to adapters installed in the administration domain and likewise assign process archives to process engines.
3. Specify the startup options for each adapter service.

Password Handling

At design time, the adapter uses a password to connect to the back-end application and fetch metadata. At runtime, the adapter uses a password to connect to the back-end application and interoperate with it. If you create a 4.x configuration using TIBCO Designer 5.1.2, and use the configuration against a 4.x adapter version, some special considerations are required for security.

If you plan to run the adapter locally, define the runtime password value to be a global variable. Before starting the adapter, include the runtime password as a client variable in the adapter's TRA file and obfuscate it using the obfuscate tool. For example, if the password value is defined as `%%myPassword%%`, create a global variable named `myPassword` in the global variables section with no value and include the following entry in the adapter's TRA file:

```
tibco.clientVar.myPassword
```

If you plan to deploy the adapter using TIBCO Administrator Enterprise Edition 5.1, checkmark the `Service` property of the global variable in the global variables section. Before deploying the adapter, go to the Advanced tab of the adapter archive and set the password value under the `Runtime Variables` section.

See Also

See the *TIBCO Administrator User's Guide* for an introduction to the TIBCO administration domain and detailed information about the above steps.

Predefined Properties

Table 16 describes predefined properties.

Table 16 Predefined Properties

Property	Description
`tibco.repourl`	Identifies the absolute pathname to the Designer project or to the DAT file where the adapter configuration is defined. This is a mandatory property.
`tibco.configurl`	Specifies the location of the adapter configuration inside the project file. This is a mandatory property. ---If a relative path is specified, the adapter service is assumed to be under the default area in the project: `/tibco/private/adapter/` ---If an absolute path is specified, the adapter configuration is looked up in the project as defined by the argument.
tibco.clientVar. adkenanLogin	The user name and password used by the repository server to access the project. This is an optional property. If the property is specified in the properties file, it overrides the value in the repository.
Tibco.clientVar. adkenanPassword	
`tibco.clientVar. adkenanSecurityRealm`	A new parameter `adkenanSecurityRealm` has been added in the tra file of the adapter. This is an optional property. If the property is specified in the properties file, it overrides the value in the repository.
tibco.clientVar.<varname>	Specifies the runtime values to substitute for global variables defined in the project. This value takes precedence over the named global value set in the project. Substitution takes place only at start up. This is an optional property. If the property is specified in the properties file, it overrides the value in the repository. Append the global variable to `tibco.clientVar`, then give its value. For example: `tibco.clientVar.DirLedger=$TIBCO_HOME/adapter/adkenan/<version_num>/myledger`

Table 16 Predefined Properties

Property	Description
tibco.env.ARBORDIR	This is the location where the Kenan application is installed. This corresponds to the `$ARBORDIR` environment variable. This is a mandatory property.
tibco.env.ARBORBIN	The bin folder under `$ARBORDIR`. This contains all middleware related executables. By default, this points to `$ARBORDIR/bin`. This is a mandatory property.
tibco.env.BSDSITE	This is the location where Kenan/FX client products are installed. This corresponds to the `$BSDSITE` environment variable. The java directory under `$BSDSITE` contains middleware libraries (Java archives) and related properties files. By default, this points to the `$ARBORDIR/bsdm_site`. This is a mandatory property.
tibco.env.TUXDIR	This is the location where BEA Tuxedo is installed. This corresponds to the `$TUXDIR` environment variable. Kenan/FX middleware uses Tuxedo as the messaging layer. This is a mandatory property.
tibco.env.TUXLIBS	This is the location where the BEA Tuxedo libraries are installed. By default, this points to `$TUXDIR/lib`. This is a mandatory property.
tibco.env.TUXPATH	This is the location where the BEA Tuxedo executables (binaries) are located. By default, this points to `$TUXDIR/bin`. This is a mandatory property.
tibco.env.WSNADDR	Specifies the server where Tuxedo WSL (WorkStation) is listening. This variable takes the form `//<host IP>:<port>`. By default this is specified as `//192.168.114.225:20531`. It has to be changed accordingly. This is an optional property. This property is applicable only if the workstation address (WSNADDR) is not specified in the `tpinit.properties` file.

Table 16 Predefined Properties

Property	Description
tibco.env.FIELDTBLS32	This variable shows the list of Tuxedo FML32 files used by the Kenan/FX middleware. The values set by default are: `bali.fml, aruba.fml, ShieldWare.fml, tpadm, Usysflds`. It is strongly recommended not to change these. This is a mandatory property.
tibco.env.FLDTBLDIR32	This variable points to the location of the FML32 files. This is a mandatory property. By default, this is `$BSDSITE/config:$TUXDIR:$TUXDIR/udataobj`.
`tibco.env.KENAN_JRE_HOME`	This variable points to the location of the 64-bit JRE installation. This points to a complete JRE installation. This is a mandatory property.
`adkenan.KenanCustomCallOutFactoryName`	This variable specifies the name of the Custom Function Factory Class. This class is used to create Custom Function message handlers. This is a mandatory property only if you are using the custom function callout feature.

Starting or Stopping the Adapter

The TIBCO Administrator Application Management module allows you to start, and stop deployed applications.

To start an adapter service from the module:

1. In the Administrator GUI left pane, expand **Application Management>** *Application-Name*>**Service Instances**.
2. In the **`Service Instance`** panel, select the checkbox next to the adapter service.
3. Click the **Start Selected** button.

   The status changes from `Stopped` to `Starting up` to `Started`.
4. To stop the adapter service, click the **Stop Selected** button.

See Also

See the *TIBCO Administrator User's Guide* for more information.

Monitoring the Adapter

TIBCO Administrator offers a number of monitoring options.

- Specifying alerts and TIBCO Hawk rulebases for each machine in the domain.
- Specifying alerts and Hawk rulebases for each adapter service.
- Viewing the log for each adapter service instance.

See Also

See the *TIBCO Administrator User's Guide* for information on configuring the monitoring options.

Chapter 7 Monitoring the Adapter Using TIBCO Hawk

Read this if you have installed TIBCO Hawk and want to use TIBCO Hawk microagents through the Monitoring tab of an adapter instance. TIBCO Hawk is an optional tool from TIBCO Software Inc. that can be used in addition to the standard trace message logging method.

Topics

Overview

TIBCO Hawk is a sophisticated tool for enterprise-wide monitoring and managing of all distributed applications and systems. System administrators can use it to monitor adapters in a wide area network of any size. TIBCO Hawk can be configured to monitor system and adapter parameters and to take actions when predefined conditions occur. These actions include: sending alarms that are graphically displayed in the TIBCO Hawk display, sending email, paging, running executables, or modifying the behavior of a managed adapter.

Unlike other monitoring applications, TIBCO Hawk relies on a purely distributed intelligent agent architecture using publish or subscribe to distribute alerts. TIBCO Hawk uses TIBCO Rendezvous for all messaging and thus gains the benefits and scalability of the TIBCO Rendezvous features of publish/subscribe, subject name addressing, interest-based routing, and reliable multicast.

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

The TIBCO Hawk Enterprise Monitor consists of these components:

- **Display**—The GUI front end that displays alarms and provides editors to create rule bases, create tests, view messages, and invoke microagents to request information or initiate an action.
- **Agents**—Intelligent processes that perform monitoring and take actions as defined in rules.
- **Rulebases**—The rules that are loaded by agents to determine agent behavior.
- **Application Management Interface (AMI)**—Manages network applications via TIBCO Rendezvous and supports communication between a network application and monitoring TIBCO Hawk agents, including the ability to examine application variables, invoke methods, and monitor system performance.
- **Microagents**—Feed information back to TIBCO Hawk and expose action methods to rulebases.

  For more information, see the TIBCO Hawk documentation.

Starting TIBCO Hawk Software

The TIBCO Hawk agent can be configured to start automatically during the system boot cycle. See the *TIBCO Hawk Installation and Configuration* guide for information about starting TIBCO Hawk.

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

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

The Auto-Discovery Process

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

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

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

Icon colors change to reflect the highest level of alert found on discovered agents. For explanations of icon elements and characteristics, see your *TIBCO Hawk Administrator's Guide*.

Invoking Microagent Methods

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

To invoke a microagent method:

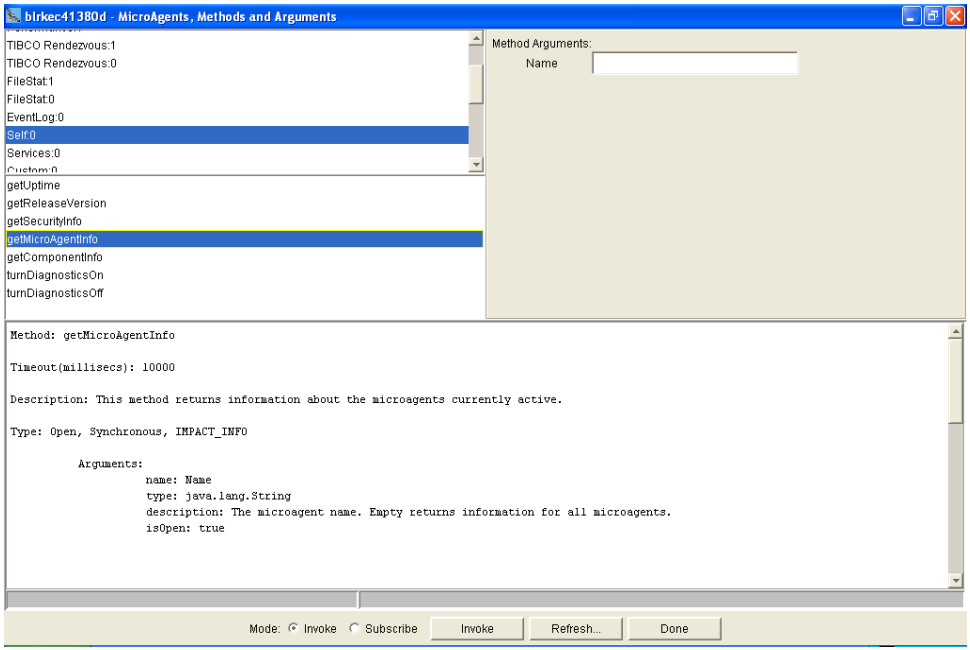
1. Start TIBCO Hawk Display and then right-click the agent icon and select **Get Microagents**.

   If TIBCO Hawk security is implemented on your system and you do not have access to microagents on this agent, an error dialog displays. Select another agent, or contact your system administrator to obtain access. The Microagents, Methods and Arguments dialog displays. The panel on the upper left lists microagents you can access on the current agent.

   This dialog has two modes, `Invoke` and `Subscribe`. Invoking a method immediately returns a single set of current results. Subscribing provides updates of current results at regular intervals. Radio buttons at the bottom of the dialog control these modes.

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

Figure 11 Getting a Microagent Information



3. Click the name of the method to invoke, such as **getComponentInfo**.

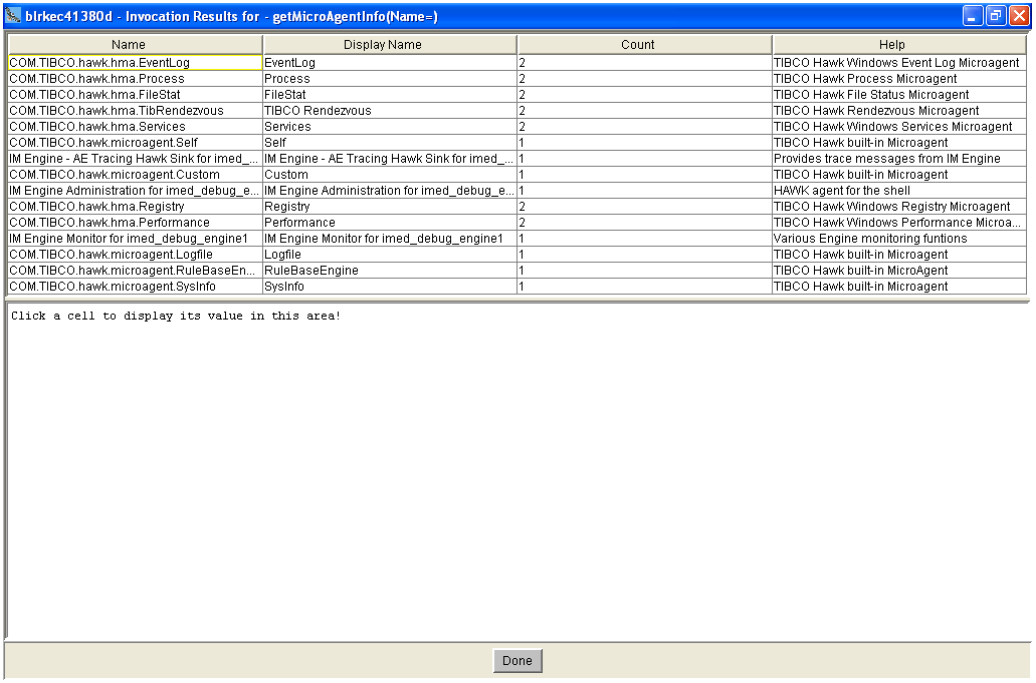
   If the method accepts arguments, fields for each argument display in the upper right panel. Detailed help text displays in the lower panel.

4. Specify any arguments for the method invocation.

5. Verify that the Invoke radio button is selected.

6. Click the **Invoke** button to invoke the selected method. The Invocation Results dialog displays the results returned by the method.

Figure 12 Invoke a Method



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

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

Available Microagents

Each adapter has two microagents, a standard TIBCO Hawk microagent named `COM.TIBCO.ADAPTER.`*xyz* (where *xyz* is the adapter configuration name) and a class microagent. These microagents provide:

- Business level statistics that report the progress of the adapter as it interacts with the database. For example, in a database adapter, such statistics might indicate whether objects were successfully or unsuccessfully inserted, updated, or deleted in the database.
- Queries that return information about the state of the adapter. This can be an important tool for seeing the internals of an adapter and debugging it if something appears wrong. For example, methods can return information about threads, internal queues, or connections to the target system. Using these methods, one might be able to identify bottlenecks or gauge how successfully an adapter is scaling with respect to the current environment.
- Updates of the adapter runtime parameters. This includes retrieving the current runtime parameters and setting new runtime parameters without restarting the adapter. An example of this is getting and setting the polling interval. Updating a runtime parameter through the Hawk microagent only affects the setting of the instance that is running. It does not permanently change the setting in either the repository or the `.tra` file.

The following table lists each method available for the adapter and the page on which the method is explained. Although the Microagents, Methods, and Arguments dialog in TIBCO Hawk Display lists more methods than are documented here, only the following methods are supported.

Table 17 Standard Microagent Methods

Method	Description	Page
`activateTraceRole()`	Activates mapping of a role to a sink at runtime.	97
`deactivateTraceRole()`	Deactivates mapping of a roles to sinks at runtime.	98
`getActivityStatistics ByService()`	Returns information about services implemented by this adapter.	99
`getAdapterServiceInfo rmation()`	Returns information about services implemented by this adapter.	100

Table 17 Standard Microagent Methods (Cont?)

Method (Cont?)	Description (Cont?)	Page
getComponents()	Returns information about the publisher, subscriber, and IODescriptor.	101
getConfig()	Returns basic configuration information. More specific information is accessed by more specific methods.	102
getConfigProperties()	Returns all attributes and elements for the given repository object.	103
getConnectionStatistics()	Returns the state and statistics for all the current connections used by the adapter.	104
getHostInformation()	Returns standard and extended application information.	105
getQueueStatistics()	Returns the current count of elements in any internal queue used by the adapter.	106
getRvConfig()	Returns information about all TIBCO Rendezvous sessions defined.	107
getStatus()	Returns general status information, such as the number of TIBCO Rendezvous messages received and published, the number of errors since the last call, the PID of the application, and more.	108
getThreadStatistics()	Returns the operation counts of the current threads.	109
getTraceSinks()	Returns information about sinks to which traces currently go.	110
getVersion()	Returns the configuration ID, application name, version, and date for this adapter instance.	111
_onUnsolictedMsg()	Displays alert messages sent to the current adapter.	112
preRegisterListener()	Preregisters an anticipated listener.	113
resetActivityStatistics()	Resets all counts for activity statistics.	114
resetConnectionStatistics()	Resets all counts for connection statistics.	115

Table 17 Standard Microagent Methods (Cont?)

Method (Cont?)	Description (Cont?)	Page
resetThreadStatistics()	Resets the counts for thread statistics.	116
`reviewLedger()`	Returns information retrieved from the ledger file of a certified messaging session for a publisher adapter.	117
`setDebugLevel()`	Sets the debug level for the current adapter instance.	119
setTraceSinks()	Adds a role or changes the file limit of a previously specified sink.	120
stopApplicationInstance()	Stops the running adapter instance.	121
`unRegisterListener()`	Unregisters a currently preregistered listener.	122

activateTraceRole()

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

Input Parameters	Type	Description
Role Name	string	Name of the role to activate.
Sink Name	string	Name of the sink for which to activate the role.

deactivateTraceRole()

Deactivates a mapping of a roles to sinks at runtime.

Input Parameters	Type	Description
Role Name	string	Name of the role to activate.
Sink Name	string	Name of the sink for which to activate the role.

getActivityStatisticsByService()

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

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

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

getAdapterServiceInformation()

Returns information about services implemented by this adapter.

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

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

getComponents()

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

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

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

getConfig()

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

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

getConfigProperties()

Returns all attributes and elements for the given repository object.

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

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

getConnectionStatistics()

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

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

getHostInformation()

Returns standard and extended application information set. It returns the following information.

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

getQueueStatistics()

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

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

getRvConfig()

Returns information about the TIBCO Rendezvous session defined by this adapter. Information about all currently defined sessions is returned if no `sessionName` is provided.

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

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

getStatus()

Retrieves basic status information about the adapter.

This information is fairly limited; for more detail, additional methods are provided (*getConfig() on page 102* and getRvConfig() on page 107.)

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

getThreadStatistics()

Return the operation counts of the current threads.

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

getTraceSinks()

Returns information about sinks to which traces currently go.

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

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

getVersion()

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

Returns	Description
Instance ID	Configuration ID as a string, for example `SDK`.
Adapter Name	Name of the adapter as a string, for example `agentone`.
Version	Version number as a string, for example `5.1`.

_onUnsolictedMsg()

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

preRegisterListener()

Preregister an anticipated subscription service. Some sending applications can anticipate requests for certified delivery even before the listening applications start running. In such situations, the publication service can preregister subscription services, so TIBCO Rendezvous software begins storing outbound messages in the publication service ledger. If the listening correspondent requires old messages, it receives the backlogged messages when it requests certified delivery.

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

Returns OK if the subscription service was preregistered successfully, false otherwise.

resetActivityStatistics()

Resets all the counts for the activity statistics and when `getActivityStatistics()` is invoked, default values are displayed.

resetConnectionStatistics()

Resets all the counts for the connection statistics and when `getActivityStatistics()` is invoked, default values are displayed.

resetThreadStatistics()

Resets all the counts for the thread statistics and when `getActivityStatistics()` is invoked, default values are displayed.

reviewLedger()

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

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

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

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

Returns (Cont?)	Type	Description
UnacknowledgedMessages	integer	Number of RVCM messages pending for this listener. The value is computed by subtracting the last sent sequence number from the last acknowledged sequence number.

setDebugLevel()

Sets the debug level for the current adapter instance.

Input Parameter	Type	Description
DebugLevel	integer	Sets the debug level to 0 (off), 1, 2, or 3. 0— No debug information displayed. 1—SQL commands executed against the database shown. 2—ODBC data source for each SQL command shown. 3—All debug information displayed.

Returns OK if successful or an error if not successful.

setTraceSinks()

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

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

Returns OK if successful or an error if not successful.

stopApplicationInstance()

Stops the specified adapter by calling the internal `stop()` method. This method returns OK if successful or an error if not successful.

In some cases, a "Method Invocation Succeeded" message appears, but the adapter does not return control to the command line.

If the `com.csgsystems.aruba.connection.PoolableATMI.cleanup` property is set to true in `pools.properties`, there will be a separate thread to do some cleanup for the Kenan connection pool. If this thread is not closed while shutting down adapter, then when invoking the `stopApplicationInstance` method using TIBCO Hawk, the adapter does not return control to command line.

If the `com.csgsystems.aruba.connection.PoolableATMI.cleanup` property is set to false, the adapter can stop and return control to command line successfully.

unRegisterListener()

Unregister a currently preregistered subscription service.

Input Parameters	Type	Description
Session Name	string	Name of the session that anticipates the subscription service.
Publisher Name	string	Name of the publication service to which the subscription service is preregistered.
Listener Session Name	string	Name of the subscription service to unregister.

This method returns true if the subscription service was unregistered successfully, false otherwise.

Appendix A Frequently Asked Questions

This appendix lists the answers to the frequently asked questions.

Topics

Frequently Asked Questions

Here are the questions:

Can I bring up TIBCO Designer from a UNIX command-line?

No. TIBCO Designer is a GUI based tool and a UNIX GUI environment is mandatory to run it. It cannot be brought up from a terminal.

When starting the adapter, what if the repository is not found?

Start the TIBCO Repository server before starting the adapter. If you are starting a remote repository ensure that TIBCO Repository is installed on the remote location. Ensure that a properly configured `.dat` file is available in the path specified (local or remote). Ensure that the `RepoUrl` has been specified accurately in the adapter's TRA file.

Why does the adapter startup fail?

Either the repository file (the DAT file) is not placed in the *TIBCO_HOME*`\repository\remoterepos` directory, or it is not properly configured. Ensure that the RepoUrl syntax has been specified accurately in the adapter's TRA file. Ensure that the path specified for the TRA file is correct.

Why does the adapter startup fail, even after specifying the appropriate DAT file?

You must start the repository server before you start the adapter. If it is a remote repository, ensure that the RepoUrl syntax has been specified accurately in the adapter's TRA file. Ensure that the path specified for the TRA file is correct.

Why is the adapter unable to discover any domains on the network even though the TIBCO ActiveMatrix BusinessWorks administration server is up and running?

Re-start it and re-discover it.

When saving an adapter configuration to the project, if an error occurs, where is it logged?

TIBCO Designer error messages are logged to the files `stderr.log` and `designer.log` under the *TIBCO_HOME*`\Designer\`*version_num*`\logs` directory.

Why does the adapter fail to respond to a request?

The subject name may be inconsistent. The subject name to which the adapter listens may be different from that of the subject name of the client.

Why does the adapter fail to respond to a request after successfully receiving it?

The adapter may fail to respond due to various reasons like errors resulting from class mismatch, records not being available in the target application or, connectivity problems with the target application.

When running the adapter, why does the following error occur?

```
Connection to TIB CR failed
```

You have referred an incorrect .dat file in your command line.

Can the adapter handle messages that include special non-English charachters such as special German or Chinese?

The adapter generally cannot handle requests that include special characters such as German. However, the current version of the audiotapes supports these special characters. When using these special characters, the user needs to correctly configure the Tuxedo Oracle connection encoding.

Please use "`UTF8`" as encoding in the Tuxedo user oracle connection, such as "`AMERICA_AMERICAN.UTF8`".

During reconnection, why does the adapter sometimes display a connection timeout error and shut down?

There is a limitation in the Kenan/BP client libraries where after a Tuxedo service reboot, used connections can not be reused anymore, so the number of valid connections in the pool decrease. As a result, the adapter cannot refresh a connection after the server goes down and is brought back up.

The adapter shutting down can be delayed by increasing the number of connections in the `pools.properties` file and by using fewer threads (proportionate to the connection pool size). For example, if the number of threads configured in the adapter is **`4`** and the connection pool size is **`40`**, the adapter can perform **`10`** reconnection attempts successfully. The connection pool size can be set using the `PoolableATMI` parameter in the `pools.properties` configuration file. Refer to the *Kenan/BP System Administration Guide* for details.

When invoking the stopApplicationInstance method using TIBCO Hawk, why does the adapter not return control to the command line, in spite of a "Method Invocation Succeeded" message

appearing?

If the `com.csgsystems.aruba.connection.PoolableATMI.cleanup` property in the `pools.properties` file is set to **`true`**, there will be a separate thread to do cleanup for the Kenan connection pool. If this thread is not closed while shutting down adapter, then when invoking the `stopApplicationInstance` method using TIBCO Hawk, the adapter does not return control to the command line. If the `com.csgsystems.aruba.connection.PoolableATMI.cleanup` property is set to **`false`**, the adapter can stop and return control to the command line successfully.

Appendix B Trace Messages

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

Topics

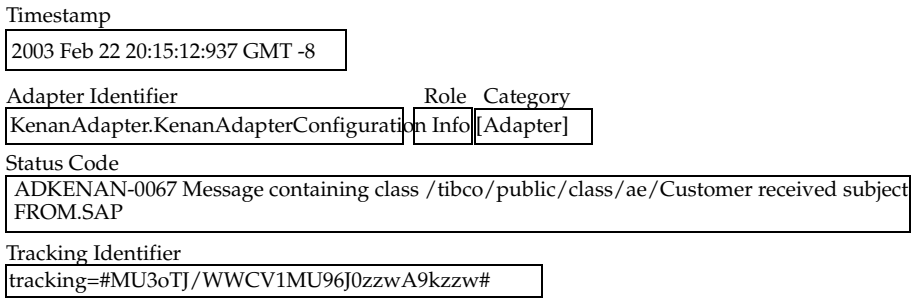
Overview

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

Each trace message can include the following fields:

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

The above fields are explained in Trace Message Fields. The following diagram shows an example trace message and calls out the fields.



Example Trace Messages

The following trace messages were written during a session where TIBCO Adapter for Kenan/BP received an object from TIBCO Adapter for R/3 and then processed the object.

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

```
2003 Apr 07 16:10:38:446 GMT +5
Kenan/BPAdapterConfiguration Info [Configuration]
ADKENAN-990038 "Application Ready".
```

The next set of trace messages indicates the adapter received an object that was sent on the TIBCO Rendezvous subject, `FROM.SAP`. The `#MU3oTJ/WWCV1MU96J0zzwA9kzzw#` tracking identifier included in the trace message uniquely identifies the message. The adapter (TIBCO Adapter for R/3) from which the message originated provided the identifier.

```
2003 Apr 07 16:14:53:943 GMT +5
KenanAdapterConfiguration Info [Adapter]
ADKENAN-990034 Incoming event is BusCompPubEvent with Key Name/A*
and Operation type 4.
tracking=#iGQSCYoeNvds1kkG67zzw6R-zzw#

2003 Apr 07 16:14:57:959 GMT +5
KenanAdapterConfiguration Info [Adapter]
ADKENAN-990036 Event BusCompPubEvent completed with result 0. Time
elapsed: 3996 ms.
tracking=#iGQSCYoeNvds1kkG67zzw6R-zzw#
```

The final trace message states that the event has been completed with result `0`.

Trace Message Fields

Each trace message includes the following fields:

Table 18 Tracing Fields

Field Name	Description
Timestamp	Timestamp of occurrence. For example, `2003 Feb 22 20:14:51:718 GMT -8`.
Adapter Identifier	This is the name of the adapter instance. For example, `KenanAdapterConfiguration`.
Role	A role can be: ---**Info**. Indicates normal adapter operation. No action is necessary. A tracing message tagged with Info indicates that a significant processing step was reached and has been logged for tracking or auditing purposes. Only info messages preceding a tracking identifier are considered significant steps. ---**Warn**. An abnormal condition was found. Processing will continue, but special attention from an administrator is recommended. ---**Error**. An unrecoverable error occurred. Depending on the error severity, the adapter may continue with the next operation or may stop altogether. ---**Debug**. A developer-defined tracing message. In normal operating conditions, debug messages should not display. When configuring the adapter, you define what roles should or should not be logged. For example, you may decide not to log Info roles to increase performance.
Category	One of the following: ---Adapter. The adapter is processing an event. ---Application. The adapter is interacting with the Kenan/BP system. ---Configuration. The adapter is reading configuration information. ---Database. The adapter is interacting with a database. ---DTA. (Design-time adapter) The trace message if from the DTA. ---Palette. The adapter is interacting with the palette. ---Request-Response Server. The Request-Response Service is reporting ---this trace message. ---Shutdown. The adapter is shutting down. ---Startup. The adapter is starting. ---Subscription Service. The Subscription Service is reporting this trace message. ---System. This category is not linked to a specific event process. ---TibRvComm. The adapter is communicating with TIBCO Rendezvous. ---XML. The adapter is parsing XML documents.

Table 18 Tracing Fields

Field Name	Description
Status Code	Unique code for the message and description. Status codes are identified by a unique number and description. If a trace message includes an error or warn role, the status code documentation includes a resolution. See Status Messages on page 131 for details.
Tracking Identifier	A unique identifier that is "stamped" on each message by the originating adapter. The tracking identifier remains in effect from a message's beginning to its completion as it is exchanged by TIBCO applications. If the adapter is the termination point of the message, the tracking identifier is not displayed in the trace message. You cannot modify the tracking identifier format or configure what information is displayed.
Application Information	Application-specific information added to the tracking info to trace the message back to its source. Set initially by the originating adapter and carried forward. It is augmented by each intermediate component.

Status Messages

Resolutions are provided wherever possible for error and warning messages. If there is no resolution provided, or if you need additional help, contact TIBCO Support at http://support.tibco.com.

Table 19 Error Messages

Message	Role	Category	Resolution
AEKENAN-710001	**Mandatory Field Error/No value specified for mandatory field [%1]**		
	Error	Configuration	Please specify the value for this mandatory field.
AEKENAN-710004	**Invalid terminate subject specified for the adapter instance.**		
	Error	Configuration	The terminate subject is syntactically invalid. Please specify a subject with a valid syntax and naming conventions.
AEKENAN-710006	**Exception in setting the model for the instance.**		
	Error	Configuration	Delete the instance and add a fresh instance. If the error still persists, restart TIBCO Designer.

Table 19 Error Messages

Message	Role	Category	Resolution
AEKENAN-710007	**Field: Terminate Subject**		
	Error	Configuration	Please specify a non-null terminate subject.
AEKENAN-710008	**Palette error. Terminate Subject field is mandatory.**		
	Error	Configuration	Please specify a non-null terminate subject.
AEKENAN-710009	**Invalid Instance Name/Adapter Configuration names must have only alphanumeric characters and can be up to 80 characters long. Please type in a valid name.**		
	Error	Configuration	Please specify a valid instance name that is alphanumeric and is less than 80 characters long.
AEKENAN-710013	**Field: Max No of Retries/Palette error. The Maximum Number of Retries must be greater than or equal to -1, and less than or equal to 65535.**		
	Error	Configuration	Please enter a valid max retries value.
AEKENAN-710014	**Invalid Service name/Service name must have only alphanumeric characters and can be up to 80 characters long. Please type in a valid name.**		
	Error	Configuration	Please enter a service name that is alphanumeric and less than 80 characters.
AEKENAN-710015	**Field: Sleep Between Retries/Palette error. The Sleep Between Retries must be greater than or equal to 0, and less than or equal to 65535.**		
	Error	Configuration	Please enter a proper numeric value for sleep between retires.
AEKENAN-710018	**Exception in adding the resource to the designer document.**		
	Error	Configuration	Please delete the instance and add a fresh instance.If the error still persists,restart Designer.
AEKENAN-710021	**Resource Rename operation Failed**		
	Error	Configuration	Check if the project is checked into a source control system.

Table 19 Error Messages

Message	Role	Category	Resolution
AEKENAN-710022	**Resource Delete operation failed**		
	Error	Configuration	Check if the project is checked into a source control system.
AEKENAN-710025	**Resource Move Operation failed**		
	Error	Configuration	Check if the project is checked into a source control system.
AEKENAN-710026	**Resource Paste operation Failed**		
	Error	Configuration	Check if the project is checked into a source control system.
AEKENAN-710027	**Warning:Add File to RCS**		
	Warning	Configuration	Check if the project is checked into a source control system.
AEKENAN-710028	**"Warning: "%1" was created during schema generation.**		
	Warning	Configuration	Check if the project is checked into a source control system.
AEKENAN-800001	**Caught an exception.Message: TPEINVAL - invalid arguments given**		
	Error	Adapter	This message is used to catch any exception thrown during adapter start-up as well as while processing messages. Hence there is now explicit resolution for the same.
AEKENAN-800002	**Caught an SDK Mexception.Message: %1**		
	Error	Adapter	Refer to the SDK documentation regarding the error thrown.
AEKENAN-800003	**Caught a Tuxedo Exception.Message: %1**		
	Error	Adapter	Check whether the Tuxedo settings provided (in `tpinit.properties`) is correct and whether the Tuxedo middleware services are up.

Table 19 Error Messages

Message	Role	Category	Resolution
AEKENAN-800004	**Exception in Reply from Kenan.Message %1**		
	Error	Adapter	Check whether the values provided in the incoming request to the adapter is correct.
AEKENAN-890001	**Unable to read Security Server Connection parameters from the repository**		
	Error	Adapter	Check whether the Security Server authentication parameters has been provided in the repository.
AEKENAN-890002	**Unable to get default context from tpinit.properties file**		
	Error	Adapter	Check the contents of the `tpinit.properties` file (default location `$ARBORDIR/bsdm_site/java`).
AEKENAN-890003	**Successfully read the default BSDMSessionContext from tpinit.properties file**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-890004	**Unable to create Security Manager Object**		
	Error	Adapter	Check whether `FXSecFwkClient.jar` is present in the classpath (default location `$ARBORDIR/bsdm_site/java`).
AEKENAN-890005	**Successfully connected to the Security Server using Login %1 and Password %2**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-890006	**Failed to connect to the Security Server using Login %1 and Password %2**		
	Error	Adapter	Check the Security Server authentication parameters.
AEKENAN-890007	**Successfully created an XML Connection**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-890008	**Failed to create an XML Connection. Exception: %1**		
	Error	Adapter	Check if the Tuxedo middleware services are running.

Table 19 Error Messages

Message	Role	Category	Resolution
AEKENAN-890010	**Failed to read the default BSDMSettings from tpinit.properties file**		
	Error	Adapter	Check the contents of the `tpinit.properties` file (default location `$ARBORDIR/bsdm_site/java`).
AEKENAN-890011	**Failed during Connection retry. Exception: %1**		
	Error	Adapter	Check if the Tuxedo middleware services are running.
AEKENAN-890012	**Failed to read the Security Server password form the repository.**		
	Error	Adapter	Check if Security Server authentication parameters have been provided in the repository.
AEKENAN-890014	**Failed to create connection factory instance**		
	Error	Adapter	Check whether the Kenan Jars (default location `$ARBORDIR/bsdm_site/java`) are in the `classpath`.
AEKENAN-890016	**Failed to establish connection with the Kenan/FX middleware after %1 attempts**		
	Error	Adapter	Check if the Kenan Security Server is Up and Middleware processes are running.
AEKENAN-890017	**Unable to create XML Connection, trying to reconnect with the application**		
	Error	Adapter	Check if the Kenan Security Server is Up and Middleware processes are running.
AEKENAN-890018	**Unable to execute XML Connection call, trying to reconnect with the application**		
	Error	Adapter	Check if the Kenan Security Server is Up and Middleware processes are running.
AEKENAN-890019	**Trying to establish connection to the Kenan/FX middleware: Attempt number %1**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-890020	**Re-established the connection to Kenan/FX middleware**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-890021	**Caught a Java linking error: %1**		
	Error	Adapter	Check the JDK used.

Table 19 Error Messages

Message	Role	Category	Resolution
AEKENAN-890022	**Caught a Java Exception Initializer error: %1**		
	Error	Adapter	Check for the JDK used
AEKENAN-890023	**Failed to connect to the Kenan/FX middleware during startup**		
	Error	Adapter	Check if the Kenan/FX Middleware processes are running.
AEKENAN-910001	**Successfully initialized the adapter**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-910002	**Startup error while creating MAppProperties object**		
	Error	Adapter	Refer to the SDK documentation.
AEKENAN-910003	**Unable to bring up the adapter with configURL %1 and RepoURL %2**		
	Error	Adapter	Check the `repourl` and the `configurl`.
AEKENAN-910004	**Unable to initialize main, MException %1**		
	Error	Adapter	Refer to the SDK documentation.
AEKENAN-910006	**Unable to initialize main, The libraries %1 are not found in the classpath**		
	Error	Adapter	Check if all the Kenan and TIBCO related jar files are in the classpath.
AEKENAN-910007	**Initializing the adapter**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-910008	**Unable to read the repoURL**		
	Error	Adapter	Check the `repourl` provided to the adapter.
AEKENAN-910009	**Unable to read the configURL**		
	Error	Adapter	Please check the `repourl` provided to the adapter.

Table 19 Error Messages

Message	Role	Category	Resolution
AEKENAN-910010	**Registering hawk methods**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-910011	**Unable to detect Request-Response Service in component registry.**		
	Error	Adapter	Check whether a Request-Response Service has been configured in the repository.
AEKENAN-910012	**Detected Request-Response Service Endpoint: %1**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-910013	**Unable to read Runtime parameters from the repository**		
	Error	Adapter	Check whether runtime parameters have been configured in the repository.
AEKENAN-910016	**Null session pointer while initializing dispatchers.**		
	Error	Adapter	Check if the multithreading tab refers to the correct session name.
AEKENAN-940003	**Unable to create the DOM document from the incoming XML message**		
	Error	Adapter	Check if the incoming request to the adapter has been configured correctly.
AEKENAN-940004	**Unable to print the DOM Document**		
	Error	Adapter	Check if the incoming request to the adapter has been configured correctly.
AEKENAN-940006	**The name of the current thread is %1**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-940007	**Unable to establish connection with application. Another thread in connection retry mode already.**		
	Error	Adapter	Check if the target application or middleware is down.

Table 19 Error Messages

Message	Role	Category	Resolution
AEKENAN-940008	**Kenans response to the request sent by the adapter**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-940009	**Request Tracking Id is %1**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-940010	**Received a request on Request-Response Endpoint**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-940011	**Successfully processed the request**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-940012	**Failed to process the request**		
	Error	Adapter	Check and correct the request sent to the Adapter
AEKENAN-940013	**Processing Header Fields : %1**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-940014	**Processing Header Fields : Header Fields values taken from %1**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-940015	**Processing Header Fields : Inserting the following nodes into the XML Message %1**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-940016	**One or all values for Header Fields not found in the XML message**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.

Table 19 Error Messages

Message	Role	Category	Resolution
AEKENAN-940017	**Detected a custom call element.**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-940018	**Extracted the custom call Class Name.**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-940019	**Starting custom call execution.**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-940020	**Finished custom call execution**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-940021	**Exception in Custom Callout: %1**		
	Error	Adapter	Check and correct the Custom Call Out Classes for the exception.
AEKENAN-940022	**The Tracking ID for this Request is %1**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-940023	**Finished processing the Request.**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-940024	**Incoming Request is %1**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-940025	**Kenan exception for the incoming request sent by the adapter is,%1'**		
	Error	Adapter	Check the request sent to the adapter.

Table 19 Error Messages

Message	Role	Category	Resolution
AEKENAN-940026	**Kenan Custom Factory Name is, %1**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-940027	**Initialized Kenan Custom Factory Class.**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-940028	**The Incoming Request Message with Headers is, %1**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-940029	**Clearing the Connection Objects**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-940030	**The custom function name is not specified**		
	Error	Adapter	Check the Custom Function name for CustomCallOut Requests.
AEKENAN-940031	**Got an invalid custom function handler**		
	Error	Adapter	Check the Custom Function name and Custom Message Handler for CustomCallOut Requests.
AEKENAN-940032	**The custom factory class name is not specified**		
	Error	Adapter	Specify the custom factory class name for CustomCall Out and ensure the custom classes are present in the classpath.
AEKENAN-990001	**Terminating the Adapter**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-990002	**Exception while terminating the adapter %1**		
	Error	Adapter	Check the Kenan Server connection parameters and app server.

Table 19 Error Messages

Message	Role	Category	Resolution
AEKENAN-990003	**Advisory Message %1**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-990004	**Advisory Message %1**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.
AEKENAN-990005	**Advisory Message %1**		
	Information	Adapter	Indicates normal adapter operation. No action necessary.

Index

M

O

P

R

S

T

U

V

X

Z