Oracle® Application Server

Adapter for Tuxedo User's Guide

10*g* Release 3 (10.1.3.1.0)

B29000-01

January 2007

ORACLE®

Oracle Application Server Adapter for Tuxedo User's Guide, 10*g* Release 3 (10.1.3.1.0)

B29000-01

Primary Author: Jeanne Wiegelmann

Contributing Authors: Yishai Hadas, Dror Harari, Adeeb Massad, Meera Srinivasan, Bo Stern, Shashi Suravarapu, Costi Zaboura, Sheela Vasudevan, Marian Jones

Contents

6 Advanced Features of OracleAS Adapter for Tuxedo

A Advanced Tuning of the Metadata

B OracleAS Adapters for Tuxedo Message Buffer Support and Data Type Support

C Advanced Tuning of the Daemon

D Globalization Settings

Index

Preface

This guide is the primary source of user and reference information on OracleAS Adapters for Tuxedo, which enables client applications to access transactions running under Tuxedo through the Sun J2EE Connector Architecture (J2CA) API.

This document describes the features of OracleAS Adapters for Tuxedo that apply to the UNIX, Windows 2000, Windows XP, and Windows Server 2003 operating systems.

This preface covers the following topics:

Audience

This manual is intended for Oracle integration administrators who perform the following tasks:

- Installing and configuring OracleAS Adapters for Tuxedo
- Diagnosing errors
- Using OracleAS to access Tuxedo transactions

> **Note:** You should understand the fundamentals of OracleAS, OC4J, the UNIX and Microsoft Windows operating system before using this guide to install or administer OracleAS Adapters for Tuxedo.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at

http://www.oracle.com/accessibility/

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Related Documents

For more information, see the following documents in the Oracle Other Product One Release 7.0 documentation set or in the Oracle Other Product Two Release 6.1 documentation set:

- *Oracle Application Server Adapter Concepts Guide*
- *Oracle Application Server Adapter Installation Guide*
- *Oracle Application Server Adapter Concepts Guide*
- *Oracle Application Server Containers for J2EE User's Guide*
- *Oracle Application Server Containers for J2EE Services Guide*
- *Oracle Application Server Containers for J2EE Security Guide*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1 Introduction

This section provides an overview of the features and architecture of OracleAS Adapters for Tuxedo.

This section contains the following topics:

OracleAS Adapters for Tuxedo Overview

OracleAS adapters for Tuxedo includes the following adapters:

- Adapter for Tuxedo
- Adapter for Tuxedo/Q

These adapters integrate Oracle Application Server with the legacy and mainframe applications. OracleAS Adapters for Tuxedo model services running on the BEA Tuxedo application server. The BEA Tuxedo application server provides two programming interfaces, Common Object Request Broker Architecture (CORBA) and Application to Transaction Monitor Interface (ATMI). ATMI supports a programming interface that offers procedural library-based programming using a set of C or COBOL procedures. ATMI also provides an interface for communication, transaction, and data buffer management. In addition, the ATMI interface and BEA Tuxedo system implement the X/Open distributed transaction processing (DTP) model for transaction processing.

Adapter for Tuxedo

OracleAS Adapter for Tuxedo includes the following features:

- Improved performance by providing direct access to the platform where OracleAS Adapter for Tuxedo runs.
- Models Tuxedo services as a collection of interactions, where each interaction is mapped to a specific Tuxedo service, along with the input and output records.
- Captures and maintains a metadata schema for the Tuxedo system by importing Tuxedo metadata either from FML and VIEW files or from a JOLT file, and transforming this metadata into mapping definitions for Oracle Connect.
- Supports Tuxedo/T synchronous calls.
- Maps Tuxedo message structures to XML data and back.

- Supports local transactions by acting as a Resource Manager (RM). This is achieved by the adapter responding to XML transactional verbs and issuing the corresponding ATMI transactional calls. The back-end service is responsible for the transaction semantics and implementation.
- Supports full client authentication with the Tuxedo server. An optional use of username and password is also supported.
- Supports the following buffer types:
  - `STRING`
  - `VIEW` (both 16-bit and 32-bit are supported)
  - `CARRAY`
  - `FML` (both 16-bit and 32-bit are supported)
  - `XML`
  - `MBSTRING` (only supported in BEA Tuxedo version 8.1 and higher)
- Supports Post events (EventBroker). This enables Java applications to subscribe to Tuxedo events, and receive them as J2CA 1.5 message inflow. Events are posted to one or several queues, according to ruling policies.
- Supports writing messages to a queue (enqueue). This is performed by executing the `enqueue` interaction for which the event is defined as the interaction input record.

OracleAS Adapter for Tuxedo does not support global transactions and cannot participate in a distributed transaction.

> **See Also:** *Oracle Application Server Adapter Concepts Guide*. BEA Tuxedo Documentation.

Adapter for Tuxedo/Q

OracleAS Adapter for Tuxedo/Q is used for pulling messages from a queue. A queue is referred to as an outbound channel, and it is logically equivalent to an outbound interaction.

To pull messages from the queue, the `getEvents` interaction is executed. This interaction is automatically added to the adapter schema. When executed, all messages are pulled from the queue. In addition, a record that unions all the output records is created and is being used as the interaction output record.

A selected masseage can be removed from the queue, using the `dequeue` interaction. When executed, a message with specific attributes is removed from the queue.

OracleAS Adapters for Tuxedo Architecture

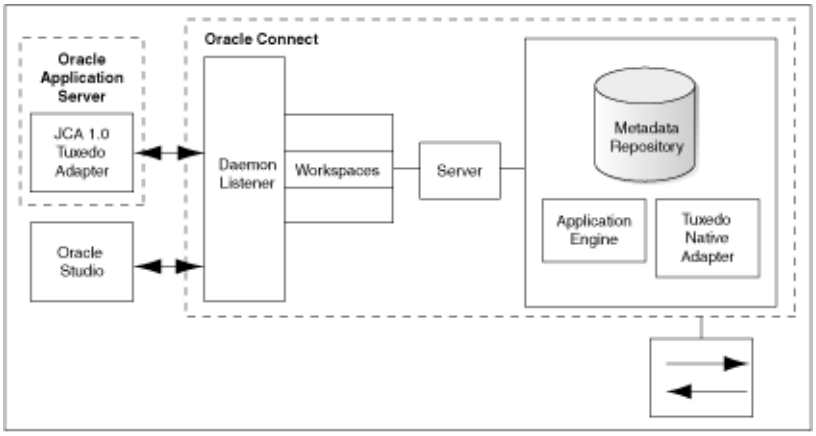
The following components comprise the architecture for Oracle AS Adapters for Tuxedo:

- J2CA 1.5 Tuxedo adapter: The J2CA Tuxedo adapter is a standard resource adapter that is compliant with J2EE Connector Architecture, providing J2EE components connectivity.
- Oracle Connect: Oracle Connect runs on the legacy system and handles requests from the J2CA 1.5 Tuxedo adapter that runs within Oracle Application Server Containers for J2EE (OC4J).

- Oracle Studio: Oracle Studio is the configuration tool for Oracle Connect. Configuration tasks using Oracle Studio are performed on a Windows platform. Oracle Studio uses perspectives that enable you to generate specific information necessary to model OracleAS Adapters for Tuxedo.

Figure 1–1 illustrates the components of OracleAS Adapter for Tuxedo.

Figure 1–1 Architecture of OracleAS Adapter for Tuxedo



See Also: *Oracle Application Server Adapter Concepts Guide*

The Flow from Oracle Application Server to the Legacy Application and Back

The J2CA 1.5 Tuxedo adapter converts the J2CA interaction invocation received from an application client to the XML format and passes the XML format to Oracle Connect on the legacy server. The daemon listens for the request coming from the J2CA 1.5 Tuxedo adapter client and assigns a server process to handle the request. The properties of the server process (such as connection pooling requirements) are defined by a workspace definition within the daemon. The server process includes an instance of the application engine, which converts the XML format into native structures understandable by Tuxedo and passes the converted XML to the back-end adapter. The back-end adapter builds an interaction based on the metadata for the back-end adapter stored in the repository and the incoming converted XML and passes it to the legacy application. The results of this execution are passed back to the application engine, using the back-end adapter, where these results are converted to XML and passed back to the client.

2

Installing and Configuring OracleAS Adapters for Tuxedo

This section describes how to install Oracle Connect and Oracle Studio from the CD-ROM, and how to configure Oracle Connect using Oracle Studio.

> **Note:** In addition to the installation tasks described in this section, the J2CA 1.5 Tuxedo adapter must be installed with Oracle Application Server. Installing the J2CA 1.5 Tuxedo adapter is described in *Oracle Application Server Adapter Installation Guide*.

This section includes the following topics:

Preinstallation Tasks

Before installing OracleAS Adapters for Tuxedo, ensure that your computer meets the following requirements:

UNIX Hardware and Software Requirements

This section describes the following requirements for installing Oracle Connect on a UNIX platform:

Hardware Requirements

The following table summarizes the hardware requirements for Oracle Connect on a UNIX platform.

Table 2–1 UNIX Hardware Requirements

Hardware Component	Requirements
Processor	HP Tru64 UNIX HP-UX IBM AIX Solaris Operating System (SPARC)
CD-ROM Drive	An internal or external CD-ROM drive
Disk Space	70 MB free disk space

Software Requirements

The following table summarizes the software requirements for Oracle Connect.

Table 2–2 UNIX Software Requirements

Software Component	Requirements
Operating System	HP Tru64 UNIX version 5.1 or higher. HP-UX version 11 or higher. **Note**: The operating system must run in 32-bit mode. IBM AIX version 4.33 or higher. **Note**: The operating system must run in 32-bit mode. Solaris Operating System (SPARC) version 2.8 or higher. **Note**: The operating system must run in 32-bit mode.
BEA Tuxedo	Version 8.0 or higher.
Oracle Application Server	Oracle Application Server 10*g* (10.1.3)

Windows Hardware and Software Requirements

This section describes the following requirements for installing Oracle Studio and Oracle Connect when OracleAS Adapters for Tuxedo run on a Windows platform:

- Hardware Requirements
- Software Requirements

Hardware Requirements

The following table summarizes the hardware requirements for Oracle Studio and Oracle Connect when OracleAS Adapters for Tuxedo run on a Windows platform.

Table 2–3 Oracle Connect Hardware Requirements on Windows

Hardware Component	Requirements
Processor	An Intel or 100% compatible personal computer (PC) based on a Pentium processor.
Memory	256 MB RAM.
CD-ROM Drive	An internal or external CD-ROM drive.
Disk Space for Oracle Studio	100 MB free disk space.
Disk Space for Oracle Connect	30 MB free disk space.

Software Requirements

The following table summarizes the software requirements for Oracle Studio or Oracle Connect when OracleAS Adapters for Tuxedo run on the Windows platform.

Table 2–4 Oracle Connect Software Requirements

Software Component	Requirements
Operating System	Microsoft Windows 2000 with service pack 2 or higher or Microsoft Windows XP, Microsoft Windows 2003.
Microsoft	Network transport protocol software, TCP/IP, included with Microsoft Windows.

Installing Oracle Connect on a UNIX Platform

This section explains how to install Oracle Connect on a UNIX platform from the CD-ROM. This section includes the following:

Preinstallation Tasks

Before starting the installation procedure, ensure that you have the following information is available:

- The root directory where you want to install Oracle Connect.

  Note: The root directory cannot be a system root directory, `/var` or `/tmp`.

- The account name where Oracle Connect will run.

Oracle Connect is present in the following operating system dependent files:

- For HP Tru64 UNIX: `ocl1012-alphaunx.taz`
- For HP-UX: `ocl1012-hpux.taz`
- For IBM AIX: `ocl1012-ibmaix.taz`
- For Solaris Operating System (SPARC): `ocl1012-sunsol.taz`

These files are provided on a CD-ROM in the `Oracle_Connect\Tuxedo_Legacy_Adapter directory`.

Installing Oracle Connect

Perform the following steps to install Oracle Connect:

1. Using FTP, copy the relevant Oracle Connect installation files in the binary mode from the installation CD-ROM.
2. Rename the taz file using the following command:

```
mv ocl1012-sunsol.taz ocl1012-sunsol.tar.Z
```

3. Decompress the file using the following command:

```
uncompress ocl1012-sunsol.tar.Z
```

4. Run the tar command, as shown in the following example:

```
tar xvf ocl1012-sunsol.tar nav_install
```

The following message is displayed:

```
x nav_install, nnnn bytes, mmmm tape blocks
```

Note: Ensure that the directory used to run the installation files has `WRITE` privileges.

Installation Tasks

Perform the following steps to install Oracle Connect:

1. Run the following command:

```
./nav_install
```

This command initiates the installation procedure. The installation procedure is displayed in a series of screen prompts and responses.

2. Enter the full path of the disk archive (.tar) file, and press **Enter**.
3. Enter the root directory name for the installation, and press **Enter**. You must have a `WRITE` permission for this directory. The default directory is the users home directory.

Notes:

- The root directory cannot be a system root directory or `/var` or `/tmp` directory.
- Oracle Connect is installed into a fixed directory named `navroot`.

---

4. Confirm the directory name in which Oracle Connect will be installed, and press **Enter**.
5. Enter the account name where you want Oracle Connect to run, and press **Enter**. This account name will be used for anonymous access to the server by clients. It can be changed after the installation is complete.
6. Confirm the account name, and press **Enter**.
7. Specify the required shell, under which Oracle Connect should run, and press **Enter**. The following options are displayed:
   - C-shell (`/bin/csh`).
   - Korn-shell (`/bin/ksh`)
   - Bourne-shell (`/bin/sh`)
8. Enter the account name for a user with administrative authorization. Optionally, press **Enter** to enable any user to administer Oracle Connect.

Postinstallation Tasks

After installing Oracle Connect, perform the following postinstallation tasks:

- Configuring the Oracle Connect Environment
- Configuring the Tuxedo Environment for Oracle Connect
- Configuring the Oracle Connect Script
- Starting the Oracle Connect Daemon

Configuring the Oracle Connect Environment

When Oracle Connect is installed on a UNIX platform, using FTP, copy the `brand.bin` file, in the binary mode, from the `Oracle Connect\Tuxedo Legacy Adapter` directory in the installation CD to the OracleAS Adapter for Tuxedo computer, to *`NAVROOT`*`/bin`.

Where *NAVROOT* is the directory where Oracle Connect is installed.

Configuring the Tuxedo Environment for Oracle Connect

Verify that the following Tuxedo environment variables are correctly set:

- `TUXDIR` is set to the Tuxedo root directory.
- `WSNADDR` is set to OracleAS Adapter for Tuxedo network address.
- Check that the shared library environment variable (`LD_LIBRARY_PATH`, `SHLIB_PATH` under HP-UX and `LIBPATH` under IBM AIX) includes the path to the Tuxedo lib directory, as in the following example:

```
LD_LIBRARY_PATH = /disk2/users/tuxedo/tuxedo8.0/lib
```

Configuring the Oracle Connect Script

The program that manages Oracle Connect server processes (`nav_server`) is accessed by a symbolic link to a file for the C-shell, Bourne and Korn shells.

To set up nav_server, perform the following steps:

1. In the `bin` directory, under the directory where Oracle Connect is installed, delete the existing link to nav_server using the following command:

   ```
   rm nav_server
   ```

2. In the `bin` directory, under the directory where Oracle Connect is installed, link to the required version of nav_server as follows:
   - C-shell: `ln -s nav_server.csh nav_server`
   - Bourne: `ln -s nav_server.sh nav_server`
   - Korn: `ln -s nav_server.ksh nav_server`

   Note: Instead of renaming files, use a symbolic link.

The Oracle Connect `nav_login` procedure defines the default environment when OracleAS Adapters for Tuxedo run. If you want site-dependent variables to be included in the environment, create a file called `site_nav_login` and save this file in the bin directory under the Oracle Connect root directory. `nav_login` runs `site_nav_login` automatically.

nav_login must be invoked to run Oracle Connect. It can be invoked from the user login script.

> **Note:** It is recommended to add TUXDIR and WSNADDR environment variables to the site_nav_login file. Adding these environment variables facilitates their availability when new server processes are started to handle client requests.

The command line for invoking nav_login varies according to the shell the user is running. The following table lists the different options for invoking the command line:

Shell	nav_login Command
CSH	.source *root*/bin/nav_login
Bourne	. *root*/bin/nav_login.sh
Korn	. *root*/bin/nav_login.sh

In the nav_login command, *root* represents the root directory of the Oracle Connect installation. After running the login procedure, the environment variable NAVROOT points to this root directory.

Ensure that users have READ and EXECUTE permissions on the Oracle Connect files. Use the chmod command to change the permissions.

Starting the Oracle Connect Daemon

The Oracle Connect daemon must run on a server for client/server access to Oracle Connect. To start the daemon with the system startup, add the following command invoking the daemon to the end of the /etc/inittab file:

```
nv:3:once:navroot/bin/irpcd -l ip:2552 start >/dev/console 2>&1
```

In this command, the symbol *navroot* should be replaced with the directory where Oracle Connect is installed and *ip* replaced by the ip address of the computer.

> **Note:** To allow automatic client/server access to Oracle Connect, start the daemon at system startup time from a super user account.

Installing Oracle Connect on a Windows Platform

This section explains how to install Oracle Connect from the CD-ROM.

Assuming that the CD-ROM drive is D:, the installation file is located in the D:\Oracle_Connect\Tuxedo_Legacy_Adapter directory. Install Oracle Connect from the CD-ROM by running the installation batch file, OCL1012-win32.bat.

> **Note:** If you are installing Oracle Connect on a Windows XP computer, you cannot specify a logical drive as the destination folder for the installation.

Installing Oracle Studio

This section explains how to install Oracle Studio from the CD-ROM.

Note: If you have Oracle Studio already installed because you are also using another legacy adapter, you do not need to reinstall it.

The other legacy adapters are:

- OracleAS Adapters for CICS (CICS and CICS Queue)
- OracleAS Adapter for IMS/DB
- OracleAS Adapter for IMS/TM
- OracleAS Adapter for VSAM

Installing Oracle Studio from the CD-ROM

Assuming that the CD-ROM drive is D:, the installation file is located in the `D:\Oracle_Studio` directory. Install the software from the CD-ROM by running the self-extracting executable installation file, `OSL904-win32.exe`.

Note: If you are installing Oracle Connect on a Windows XP computer, you cannot specify a logical drive as the destination folder for the installation.

Configuring Oracle Connect

You can configure Oracle Connect using Oracle Studio. To use Oracle Studio, first configure it to enable access to the platform where OracleAS Adapters for Tuxedo run.

Before configuring Oracle Connect, ensure the following requirements are fulfilled:

- You have permission to access the platform where OracleAS Adapters for Tuxedo runs.
- Oracle Connect daemon is running on this computer.
- Tuxedo queue space and a queue are defined.
- The Tuxedo queue allows storing persistent messages.
- The TMQUEUE server is set to handle Q-space.
- The `EventBroker` server process is set. Either `TMUSREVT` for user events, or `TMSYSEVT` for system events.

Note: For further information on BEA Tuxedo, see BEA Tuxedo documentation.

To configure Oracle Connect, refer to the following sections:

- Configuring OracleAS Adapters for Tuxedo in Oracle Studio
- Securing Access to Oracle Connect
- Modeling Interactions for OracleAS Adapter for Tuxedo
- Modeling Interactions for the OracleAS Adapter for Tuxedo Queue
- Viewing the XML Schema
- Creating XML Schemas

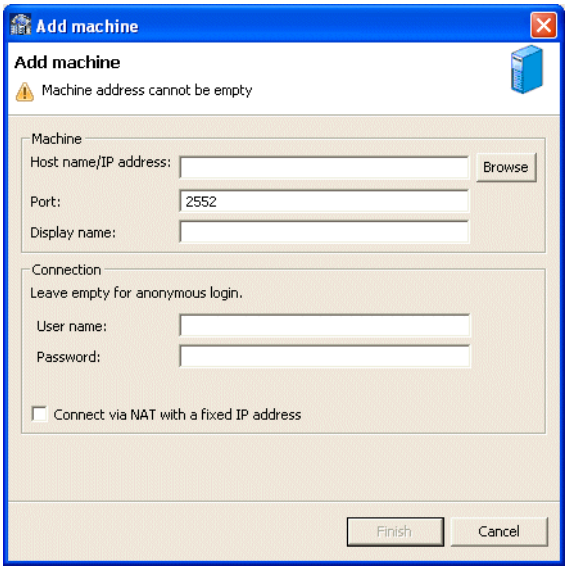
Configuring OracleAS Adapters for Tuxedo in Oracle Studio

Perform the following steps to configure OracleAS adapters for Tuxedo in Oracle Studio:

1. From the **Start** menu, select **Programs, Oracle**, and then select **Studio**. Oracle Studio opens, displaying the Design perspective and the Welcome pane.
2. Right-click **Machines** in the Configuration Explorer and select **Add Machine**.

   The Add Machine screen is displayed, as shown in the following figure:

Figure 2–1 The Add Machine screen

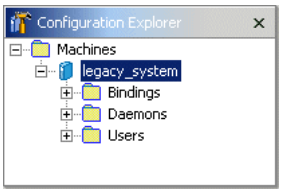


3. Enter the name of the computer you want to connect to, or click **Browse** to select the required computer that uses the default port (2552).
4. Specify the username and password of the user who was specified as the administrator during Oracle Connect installation.

> **Note:** Selecting **Anonymous connection** enables all users having access to the computer to be an administrator.

5. Click **Finish**. The selected computer is displayed in the Configuration Explorer as shown in the following figure:

Figure 2–2 The Configuration Explorer



Securing Access to Oracle Connect

Oracle Studio includes mechanisms to secure access to Oracle Connect both during design time and runtime.

During design time, the following security mechanisms can be applied:

- Setting Password Access to Oracle Studio
- Specifying Users with Administrative Rights

During runtime, client access to Oracle Connect is provided by the user profile:

- Configuring Run-Time User Access to OracleAS Adapter for Tuxedo

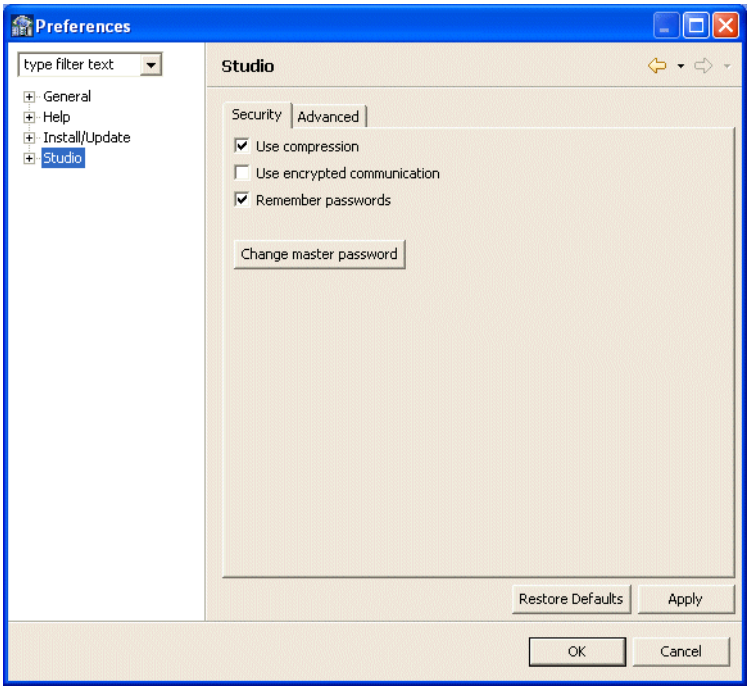
Setting Password Access to Oracle Studio

Initially, any operation performed using Oracle Studio does not require a password. You can set a password so that the first operation that involves accessing the server from Oracle Studio requires a password to be entered.

Perform the following steps to set password access to Oracle Studio:

1. From the **Start** menu, select **Programs, Oracle** and then select **Studio**. Oracle Studio opens.
2. Select **Window** from the menu bar and then select **Preferences**. The Preferences window is displayed.
3. Click the **Studio** node, as shown in the following figure:

Figure 2–3 The Preferences screen



4. Click **Change master password** to open the Change Master Password screen.
5. In the Change Master Password screen, leave the **Enter current master password** field empty and type a new master password.
6. Confirm the new password.

7. Click **OK**.

Specifying Users with Administrative Rights

By default, only the user who was specified as an administrator during the installation is authorized to modify settings on that computer from Oracle Studio. The administrator can authorize other users to make changes or view the definitions for a selected computer.

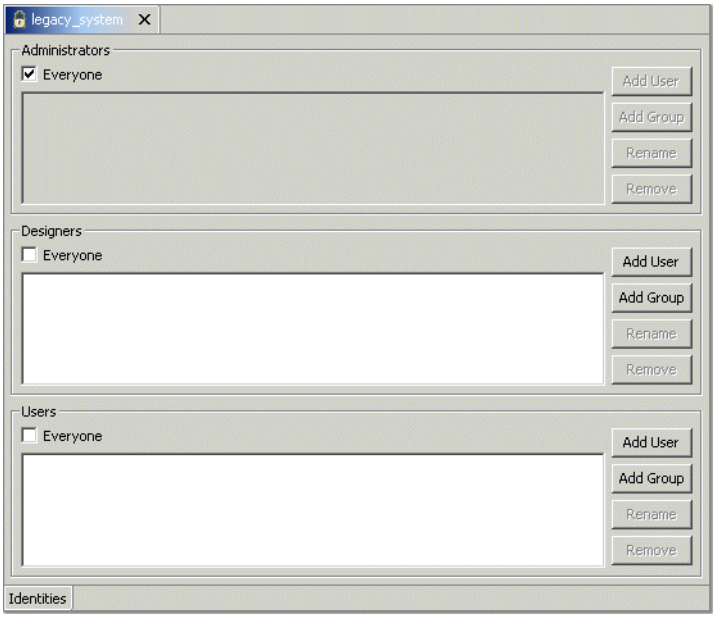
> **Note:** By default, all users should be given administrative rights during installation.

Perform the following steps to specify users with administrative rights:

1. From the **Start** menu, select **Programs, Oracle** and then select **Studio**. Oracle Studio opens, showing the Design perspective.
2. Right-click the required computer in the Configuration Explorer and select **Administration Authorization**.

   The Administration Authorization screen is displayed, as shown in the following figure:

Figure 2–4 The Administration Authorization screen



The screen has three sections:

- Administrators: Administrators can view and modify all the definitions for the selected computer in Oracle Studio. On initial entry to Oracle Studio, every user is defined as a system administrator.
- Designers: Designers can view all the definitions for the computer in Oracle Studio and can modify any of the definitions under the **Bindings** and **Users** nodes for the selected computer.

  For example, Oracle Studio database administrator can add new data sources and adapters and can change metadata definitions for a table in a data source.

- Users: Users can view all the definitions for the selected computer in Oracle Studio. Regular users cannot modify any of the definitions.

3. Add users or groups by clicking **Add User** or **Add Group** for the relevant sections.

   The user or group that is added must be recognized as a valid user or group for the computer.

> **Note:** Once a name has been added to a section, only the user or group who logs on with that user name has the relevant authorization.

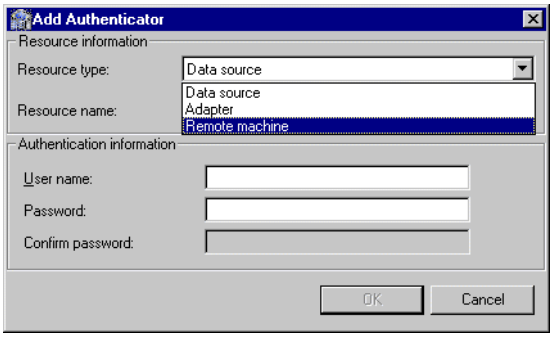
Configuring Run-Time User Access to OracleAS Adapter for Tuxedo

During runtime, client access to Oracle Connect is provided by the user profile. A user profile contains name and password pairs that are used to access a computer, data source or application during runtime, when anonymous access is not allowed.

Perform the following steps to configure runtime user access to OracleAS adapters for Tuxedo:

1. In the Configuration Explorer, expand the node of the computer which you want to set the user name and password.
2. Expand the **Users** node.
3. Right-click the **NAV** user profile and select **Edit User**. The NAV user profile editor is displayed.
4. In the User editor, click **Add** to open the Add Authenticator screen, as shown in the following figure.

Figure 2–5 The Add Authenticator screen



5. Select the **Remote Machine** resource type.
6. Enter the name of the OracleAS Adapter for Tuxedo computer defined in Oracle Studio.
7. Enter the user name and password used to access the computer and confirm the password.
8. Click **OK**.

Modeling Interactions for OracleAS Adapter for Tuxedo

Modeling interactions for OracleAS Adapter for Tuxedo involves defining an Oracle Connect back-end adapter using Oracle Studio.

All the definitions specified in Oracle Studio are written to the platform where OracleAS Adapters for Tuxedo run.

This section contains the following:

- Configuring an Oracle Connect Adapter
- Generating Outbound Interactions
- Generating Inbound Interactions

Configuring an Oracle Connect Adapter

To work with Oracle Connect, you need to configure the adapter definition on the platform where OracleAS Adapters for Tuxedo run to handle the interactions to and from a Tuxedo service.

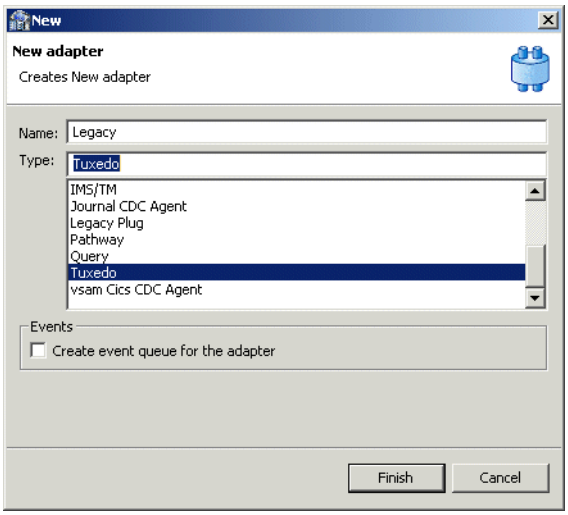
Perform the following steps using Oracle Studio to specify these definitions:

1. From the **Start** menu, select **Programs, Oracle** and then select **Studio**.
2. In the Configuration Explorer, expand the node of the computer defined in "Configuring OracleAS Adapters for Tuxedo in Oracle Studio" on page 2-8.
3. Expand the **Bindings** node. The binding configurations available on this computer are listed.
4. Expand the **NAV** binding node. The NAV binding configuration includes branches for data sources and adapters that are located on the computer.
5. Right-click **Adapters** and select **New Adapter** to open the New Adapter wizard.
6. Enter a name for the back-end adapter.

> **Note:** The word *event* is a reserved word and cannot be used to name an adapter.

7. Select **Tuxedo** as the back-end adapter type from the **Type** list, as shown in the following figure:

Figure 2–6 The New Adapter screen



8. Select **Events** to create an event queue for the adapter.

9. Click **Finish**. The back-end adapter is added to the adapters list and its definition opens for editing.

> **Note:** Other adapters that are displayed in the `Type` list are not supported with the version of Oracle Connect installed at the site.

Generating Outbound Interactions

Oracle Connect requires metadata describing the adapter interactions, including the structures used to pass information to and from the adapter.

If either BEA JOLT bulk loader file or Tuxedo configuration and FML or VIEW source files describing the adapter are available, you can import them using the Metadata Import wizard in Oracle Studio Design perspective to generate interaction metadata. If either BEA JOLT bulk loader file or Tuxedo configuration and FML or VIEW source files describing the input and output structures are not available, you need to manually define the metadata. For details of the metadata definition refer to Appendix A, "Advanced Tuning of the Metadata".

The following information is required during the import procedure:

BEA JOLT bulk loader file or Tuxedo configuration and FML or VIEW source files. These are copied to the computer running Oracle Studio as part of the import procedure.

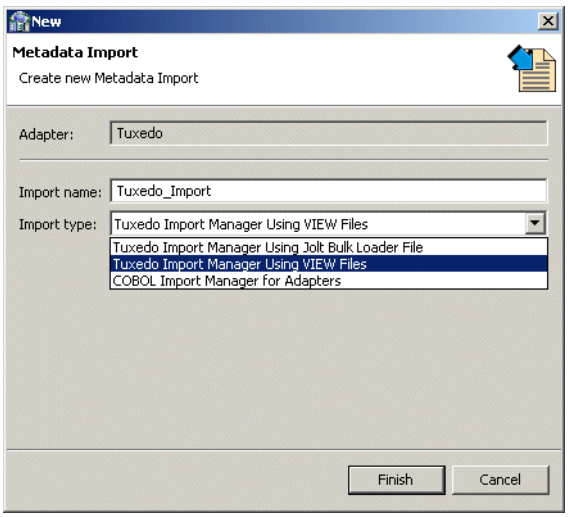
Using Oracle Studio, complete the Metadata Import steps to generate interaction metadata, as follows:

1. From the **Start** menu, select **Programs, Oracle** and then select **Studio**.
2. In the Configuration Explorer, expand the node of the computer defined in "Configuring OracleAS Adapters for Tuxedo in Oracle Studio" on page 2-8.
3. Expand the **Bindings** node. The binding configurations available on this computer are listed.
4. Expand the **NAV** binding node.
5. Expand the **Adapters** node.
6. Right-click the Tuxedo back-end adapter defined in "Configuring an Oracle Connect Adapter" on page 2-12.
7. Select **Edit Metadata** to open the Metadata tab, with the Tuxedo back-end adapter displayed under the adapters list.
8. Right-click the Tuxedo back-end adapter and select **New Import**. The New Metadata Import screen is displayed.
9. Enter a name for the import. The name can contain letters and numbers and the underscore character only.
10. Select the metadata import type. You can import the metadata either from a BEA JOLT file or from Tuxedo record definition files (FML or VIEW files).

> **Note:** A JOLT file consists of definitions of Tuxedo services along with the metadata, using the BEA JOLT Bulk Loader. `FML` and `VIEW` files are metadata files used by Tuxedo in conjunction with a configuration file that includes the Tuxedo services.

For a JOLT file, select **Tuxedo Import Manager Using Jolt Bulk Loader File**. For FML or VIEW files, select **Import Manager Using VIEW Files** as shown in the following figure:

Figure 2–7 The Metadata Import screen

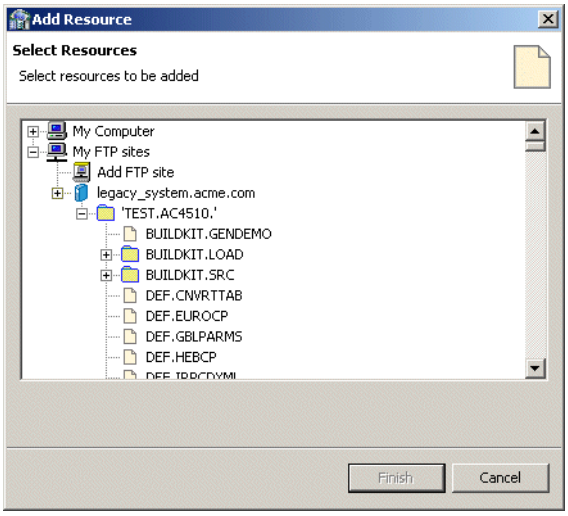


11. Click **Finish**. The Metadata Import wizard is displayed.
12. Click **Add**. The Select screen is displayed. This screen lists any files selected for use as input for the import.
13. Click **Add** in the Select screen. The Select Resources screen is displayed, which provides the option to select files from the local computer or to copy the files from another computer.
14. If the files are on another computer, right-click My FTP Sites and select Add. Optionally, double-click **Add FTP site**. The Add FTP Site screen is displayed.
15. Enter the name or IP address of the server where the JOLT bulk loader file or the FML or VIEW files reside and enter a valid username and password to access the computer (if anonymous connection is used, select Anonymous connection), then click **OK**.

    The FTP site is added to the list of available sites.

    The computer is accessed enabling you to browse and transfer files required to generate the metadata, as shown in the following figure:

Figure 2–8 The Select Resources screen

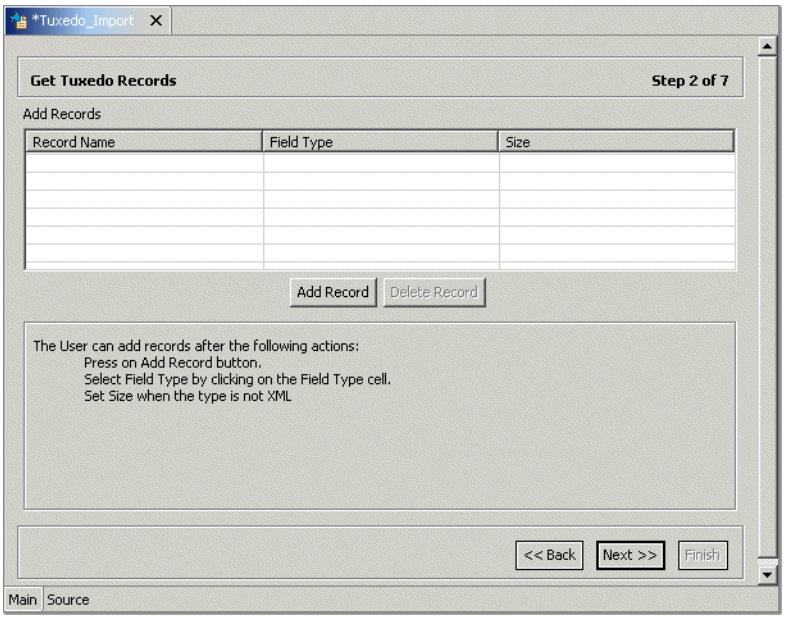


16. Right-click the required computer and select Set Transfer Type. Enter the transfer type (ASCII or BINARY) and click OK.
17. Click the node of the added site and locate the necessary file. To change the root directory, right-click the computer and select Change Root Directory. Enter the root directory and click **OK**.
18. Select the required file or files and click **Finish**. The selected file or files are displayed in the Metadata Import wizard.
19. Click **Next**. The next step of the procedure depends on whether a JOLT file or FML or VIEW files are used for the import.

    If a `JOLT` file is used, the Apply Filters screen is displayed.

    If FML or VIEW files are used, the Get Tuxedo Records screen is displayed, as shown in the following figure:

Figure 2–9 The Get Tuxedo Records screen



Use this screen to add additional simple record definitions that are not included in the FML or VIEW files used for the import. You can include information stored in the following Tuxedo buffer types:

- XML data
- String data
- Carrays

Specify the name of the record and select the buffer type from the list in the **Field Type** field.

Unstructured message buffers are wrapped within a record as follows:

- A message buffer of type `STRING` is wrapped within a record containing a single field of type string.
- A message buffer of type `CARRAY` is wrapped within a record containing a single field of type binary with a fixed size.
- A message buffer of type `XML` is wrapped within a record containing a single field of type XML.

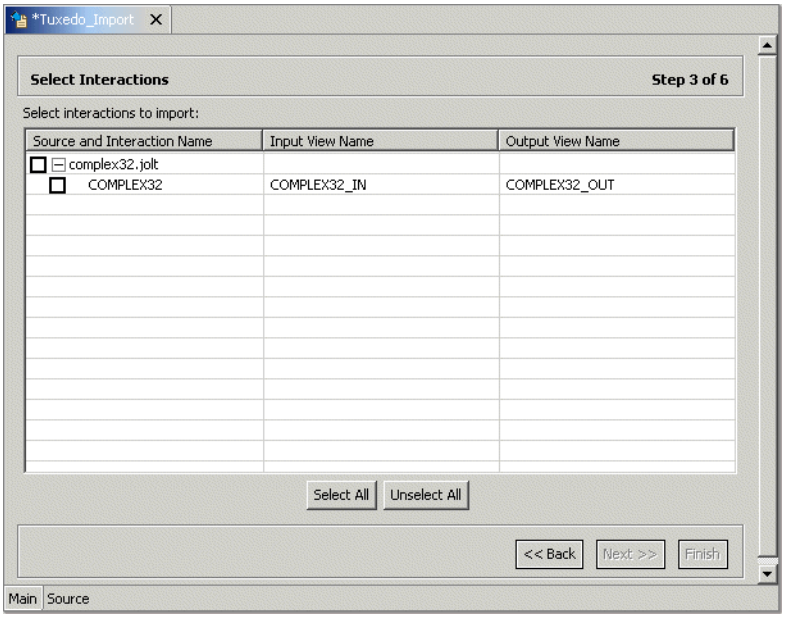
See Also: Appendix B, "OracleAS Adapters for Tuxedo Message Buffer Support and Data Type Support" for details about support for Tuxedo message buffers and data types.

20. Click **Next**. The next step in the procedure depends on whether a JOLT file, an FML or VIEW files are used for the import.

    If a JOLT file is used, the Select Interactions screen is displayed (Figure 2–10). The screen lists the services from the input JOLT file as interactions, along with the input and output structures to use with each interaction.

    Select the interactions to implement from the list, and then click Next.

Figure 2–10 The Select Interactions screen

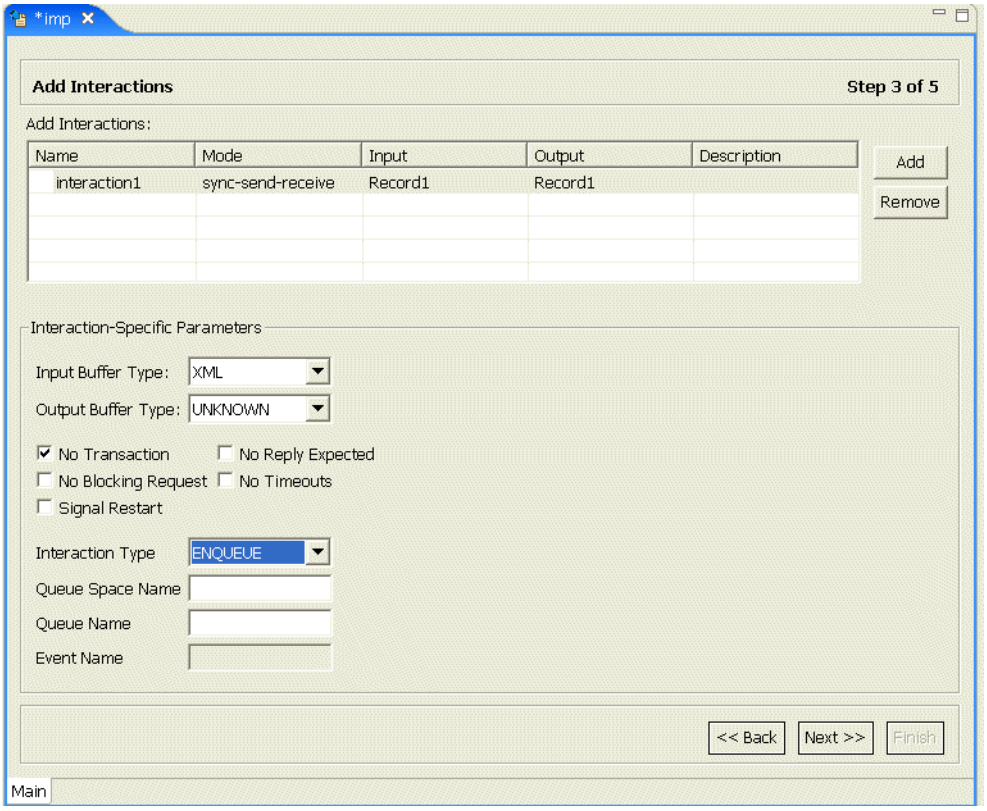


If FML or VIEW files are used, the Add Interactions screen is displayed.

21. Click **Add** to add an interaction for the Tuxedo adapter.

    The Add Interaction screen is displayed in the following figure:

Figure 2–11 The Add Interactions screen



22. Add as many interactions as required. Provide the following information for each interaction:

    - Interaction name: The name of the interaction. You can change the default name.
    - Mode: The interaction mode. You can select one of the following:
      - `sync-send-receive`: The interaction sends a request and expects to receive a response. This is the default mode.
      - `sync-receive`: The interaction expects to receive a response.
      - `sync-send`: The interaction sends a request and does not expect to receive a response.
    - Output: Identifies an output record. The output record is the data structure for the results of the interaction. The records generated from the input files specified at the beginning of the procedure are listed. Select the relevant record for the interaction.

    Note: You must specify an output record for the interaction if the mode is set to `sync-send-receive` or `sync-receive`, before you can click **Next**.

- Input: Identifies an input record. The input record is the data structure for the interaction. The records generated from the input files specified at the beginning of the procedure are listed. Select the relevant record for the interaction.

  Note: You must specify an input record for each interaction before you can click **Next**.

  If the interaction does not require an input record, the record specified here is ignored.

- Description: Free text describing the interaction.
- Interaction-Specific Parameters: Tuxedo specific parameters, as listed in the following table:

Table 2–5 Interaction-specific Parameters

Parameter	Description
`Input Buffer Type`	The type of buffer used for the input.
`Output Buffer Type`	The type of the buffer to use for the results of an outbound interaction.
`No Transaction`	Enables to run a service, regardless of transaction context. This parameter should always be checked.
`No Reply Expected`	For future use.
`No Blocking Request`	Avoids a FROM request submission if a blocking condition exists.
`No Timeouts`	Ignores blocking timeouts.

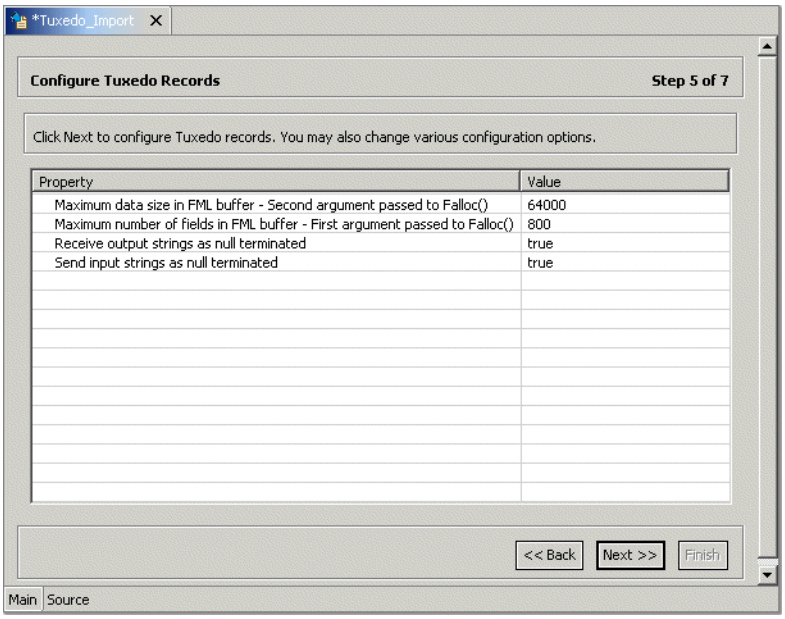
- Signal Restart: When selected, if a signal interrupts any underlying system calls, the interrupted system call is reissued.
- Interaction Type: Your options are:
  - SERVICE: Sends a request and synchronously awaits its reply. This is the default setting.
  - ENQUEUE: Places a message in the queue.

    If you select this option, you must must also fill in the fields Queue Space Name and Queue Name with the relevant values to assign the queue where the message is to be placed.
  - POST: Places a message one or more queues, according to rules and filters predefined according to events that have been subscribed to.

    If you select this option, you must also fill in the field Event Name.

23. Click **Next**. The Configure Tuxedo Records screen is displayed. It is used to specify how the Tuxedo records should be configured, as shown in the following figure:

Figure 2–12 The Configure Tuxedo Records screen



24. Click **Next** to generate the metadata definitions for the adapter.
25. Specify that you want to transfer the metadata from the Windows computer to the computer where OracleAS Adapter for Tuxedo runs, and click **Finish**.

The metadata is imported based on the options specified and it is stored on the computer where OracleAS Adapter for Tuxedo runs. An XML representation of the metadata is generated.

After performing the import, you can view the metadata in Oracle Studio Design perspective **Metadata** tab. You can also make any fine adjustments to the metadata and maintain it, as necessary.

> **See Also:** Appendix A, "Advanced Tuning of the Metadata" for details about fine tuning the adapter metadata

Generating Inbound Interactions

Inbound interactions are defined as events in Oracle Studio. An event adapter is defined automatically when you selected the Tuxedo back-end adapter with **Create event queue for the adapter** selected, as described in "Configuring an Oracle Connect Adapter" on page 2-12. The event adapter is defined with the same name as the back-end adapter with the word event appended to it.

The back-end adapter and the event adapter are linked by Oracle Studio. You can skip from the adapter definition to the event definition by right-clicking the adapter, or event, in the Configuration explorer, and selecting Linked Event or Linked Adapter, respectively.

The event adapter requires metadata describing the inbound interactions, including the structure used to passthe information.

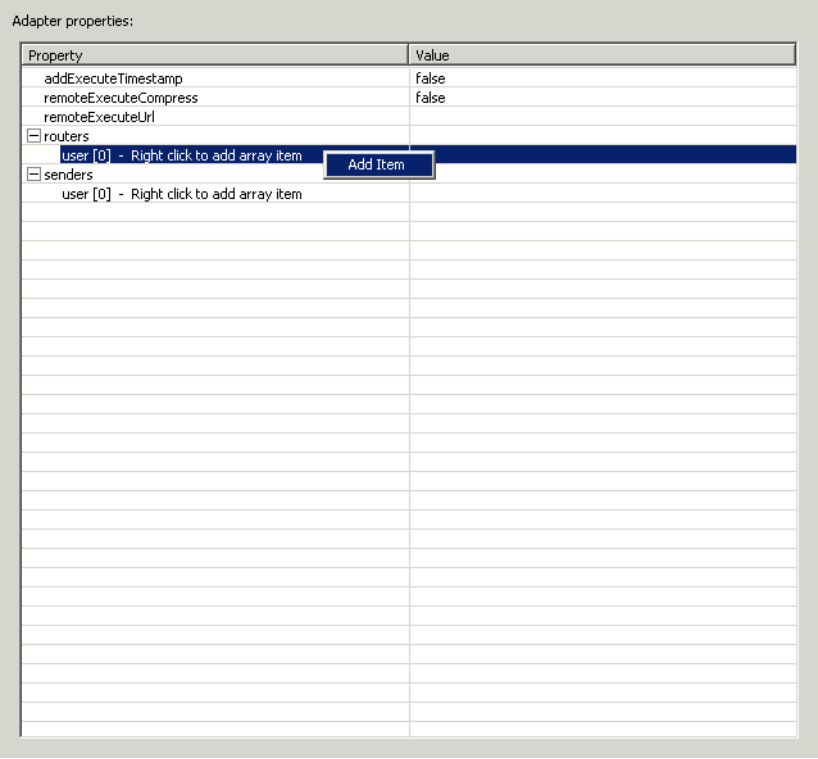
> **Note:** The generation of inbound interactions involves similar steps to the steps described to generate outbound interactions. For details, see "Generating Outbound Interactions" on page 2-13.

During the import procedure, the BEA JOLT bulk loader file or Tuxedo configuration and FML or VIEW source files are used. These are copied to the computer running Oracle Studio as part of the import procedure.

Perform the Metadata Import steps to generate inbound interaction metadata, as follows:

1. In the Configuration Explorer, right-click the Tuxedo back-end adapter defined in "Configuring an Oracle Connect Adapter" on page 2-12.
2. Select **Linked Event** to skip to the event adapter.
3. Right-click the event adapter and select **Edit Event**.
4. Click the **Properties** tab to add the names of Oracle Application Server users who can retrieve inbound interactions and users who can send inbound interactions.
5. To add Oracle Application Server users, expand the **Routers** node, right-click **user**, and then select **Add Item**, as shown in the following figure:

Figure 2–13 The Adapter Properties screen

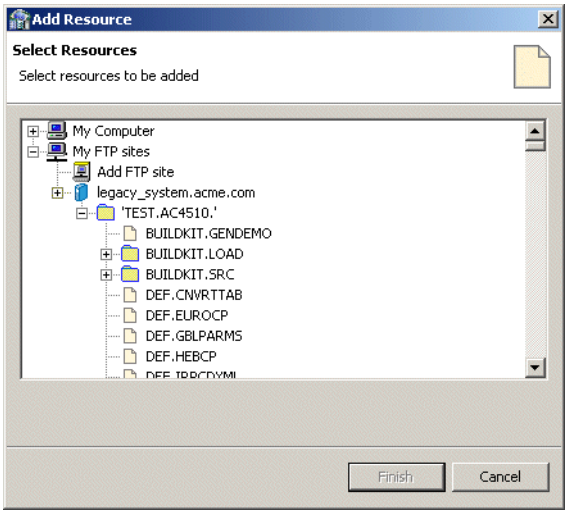


6. Enter the name of the Oracle Application Server user in the **Value** column for the item added.
7. To add users, expand the **Senders** node, right-click **Users**, and then select **Add item**. A new entry is added under the **User** node.
8. Enter the name of the user in the **Value** column for the item added.
9. Click **Save** to save the changes.
10. Right-click the event adapter in the Configuration Explorer and select **Edit metadata** to display the **Metadata** tab with the event adapter displayed under the Events list.

11. Right-click **Imports** and select **New Import**. The New Import screen is displayed.
12. Enter a name for the import. The name can contain letters, numbers and the underscore character only.
13. Select **Event Queue Import Manager Using Tuxedo View/FML Files** as the import type, and then click Finish.
14. After defining an import type, the Metadata Import wizard opens in Oracle Studio. FML or VIEW files are used to create the metadata. The Import wizard generates record structures, which are used for the record structures for inbound interactions.
15. Click **Add**.
16. The Select Resources screen is displayed, which provides the option to select files from the local computer or copy the files from another computer.
17. If the files are on another computer, right-click **My FTP Sites** and select **Add**. Optionally, double-click **Add FTP Site**. The Add FTP Site screen is displayed.
18. Enter the server name or IP address where the FML or VIEW files reside and enter a valid username and password to access the computer (if anonymous access is used, select Anonymous connection), then click **OK**. The FTP site is added to the list of available sites.

> **Note:** The selected server is accessed using the username as the high-level qualifier, enabling you to browse and transfer files.

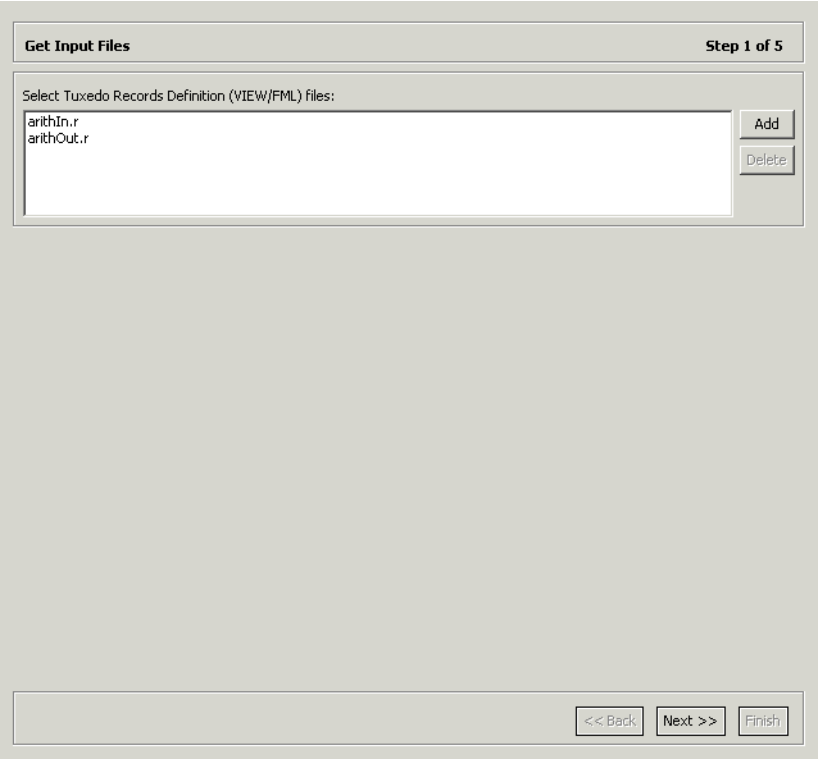
Figure 2–14 The Add Resource screen



19. Right-click the computer and select **Set Transfer Type**. Enter the transfer type (ASCII or BINARY) and click OK.
20. Expand the node of the added site and locate the necessary FML or VIEW files. To change the high-level qualifier, right-click the computer and select **Change Root Directory**. Enter the new high-level qualifier enclosed in quotes, and click **OK**.
21. Select the required FML or VIEW file or files and click **Finish**.

    The selected file or files are displayed in the Metadata Import wizard, as shown in the following figure:

Figure 2–15 The Get Input Files screen



22. Click **Next**.
23. Click **Next** to analyze and convert the selected FML or VIEW files.

    The Add Events screen is displayed.
24. Click **Add** to add an event.

    The Add Events screen is displayed.
25. Add as many interactions as required. For each interaction, provide the required information, as listed in the following table:

Table 2–6 Interaction Parameters

Property	Description
Name	The name of the interaction. You can change the default name.
Mode	The Interaction mode.
async-send	The interaction sends a request and does not expects to receive a response.
Input	Identifies an input record. The input record is the data structure for the interaction. The records generated from the FML or VIEW files specified are listed. Select the relevant record for the interaction. Note: You must specify an input record for each interaction before you can click **Next**
Description	Free text describing the interaction.

26. Click **Next** to generate the metadata definitions for the adapter.

27. Specify that you want to transfer the data from the Windows computer to the target platform, then click **Finish**.
28. The metadata is imported based on the options specified. An XML representation of the metadata is also generated. After performing the import, you can view the metadata in Oracle Studio Design perspective **Metadata** tab. You can also make any fine adjustments to the metadata and maintain it, as necessary.

    See Also: Appendix A, "Advanced Tuning of the Metadata" for details about fine tuning the adapter metadata

Modeling Interactions for the OracleAS Adapter for Tuxedo Queue

Modeling interactions for OracleAS Adapter for Tuxedo Queue involves defining an Oracle Connect back-end queue adapter using Oracle Studio.

All the definitions specified in Oracle Studio are written to the platform where OracleAS Adapter for Tuxedo Queue runs.

This section contains the following:

- Configuring the Tuxedo Queue Adapter
- Generating Inbound Interactions

Configuring the Tuxedo Queue Adapter

To work with Oracle Connect, you need to configure the queue adapter definitions on the platform where OracleAS Adapter for Tuxedo Queue runs to handle the events from a Tuxedo Queue.

Perform the following steps to specify these definitions:

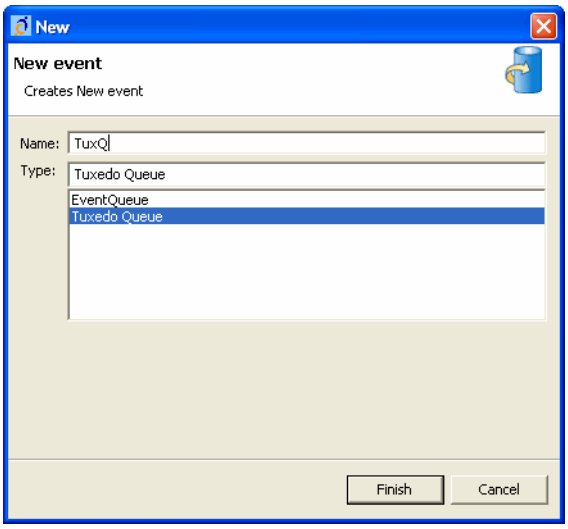
1. From the **Start** menu, select **Programs, Oracle** and then select **Studio**.
2. In the Configuration Explorer, expand the node of the computer defined in "Configuring OracleAS Adapters for Tuxedo in Oracle Studio" on page 2-8.
3. Expand the **Bindings** node. The binding configurations available on this computer are listed.
4. Expand the **NAV** binding node. The NAV binding configuration includes branches for data sources and adapters that are located on the computer.
5. Right-click **Events** and select **New Event**.

   The New Event wizard is displayed.
6. Enter a name for the event queue.

   Note: The word *event* is a reserved word and cannot be used to name an event.

7. Select **Tuxedo Queue** as the event type from the **Type** list, as shown in the following figure:

Figure 2–16 The New Event screen



8. Click **Finish**.

Generating Inbound Interactions

The queue adapter requires metadata describing the inbound interaction, including its structure.

During the import procedure, Tuxedo `FML` or `VIEW` configuration source files are copied to the computer running Oracle Studio as part of the import procedure. Alternatively, this procedure enables you to manually define the queue adapter metadata.

Before generating the interactions, note the following:

- All the events described in a single Tuxedo Queue adapter, should have the same Tuxedo buffer type.
- In case that FML/FML32 buffer type is used, a common field with the same `FBName` must be included in all events. This field should contain the record/event name.
- The interaction is of `async-send` type (it does not expect to receive a response).

Perform the following steps to generate inbound interaction metadata:

1. In the Configuration Explorer, select the Tuxedo Queue back-end adapter defined in Configuring the Tuxedo Queue Adapter on page 2-23.
2. Right-click the queue adapter in the Configuration Explorer and select **Edit metadata**.

   The **Metadata** tab is displayed, with the queue adapter displayed under the Adapters list.
3. Right-click **Imports** and select **New Import**.

   The New Import screen is displayed.
4. Enter a name for the import. The name can contain letters, numbers and the underscore character only.
5. Select the import type from the **Import Type** list. You have the following options:

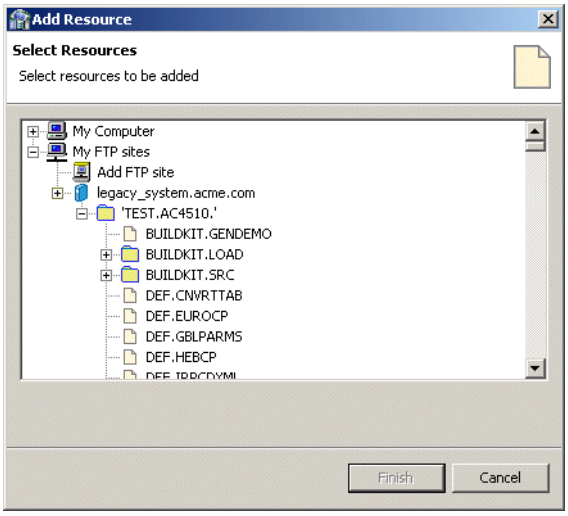
- Tuxedo Queue Import Manager for XML/STRING/CARRAY Buffers: This option does not require importing any files. It enables you to manually define the required Tuxedo record.
- Tuxedo Queue Import Manager for VIEW/VIEW32 Buffers:
- Tuxedo Queue Import Manager for FML/FML32 Buffers:

6. Click **Finish**.

   The next step depends on the selected import type. If either the `VIEW/VIEW32 Buffers` or the `FML/FML32 Buffers` options is selected, then the Metadata Import wizard opens, in which case, proceed to the following step. If the `XML/STRING/CARRAY Buffers` option is selected, proceed now to "Defining the Tuxedo Queue Unstructured Records" on page 2-27.

7. Click **Add**.
8. The Select Resources screen is displayed, which provides the option to select files from the local computer or copy the files from another computer.
9. If the files are on another computer, right-click **My FTP Sites** and select **Add**. Optionally, double-click **Add FTP Site**. The Add FTP Site screen is displayed.
10. Enter the server name or IP address where the required reside and enter a valid username and password to access the computer (if anonymous access is used, select Anonymous connection), then click **OK**. The FTP site is added to the list of available sites.

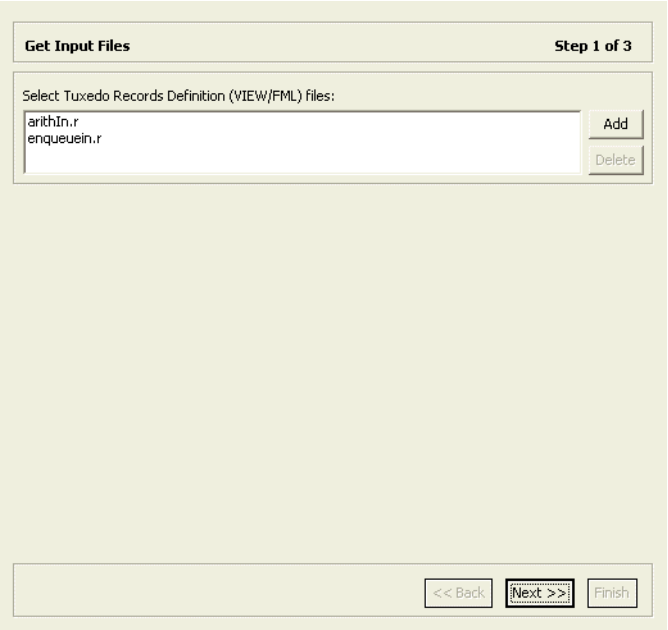
Figure 2–17 The Select Resources screen



11. Right-click the computer and select **Set Transfer Type**. Enter the appropriate transfer type, and click OK.
12. Expand the node of the added site and locate the files. To change the directory, right-click the computer and select **Change Root Directory**. Enter the new directory name, and click **OK**.
13. Select the required file or files and click **Finish**.

    The selected file or files are displayed in the Metadata Import wizard, as shown in the following figure:

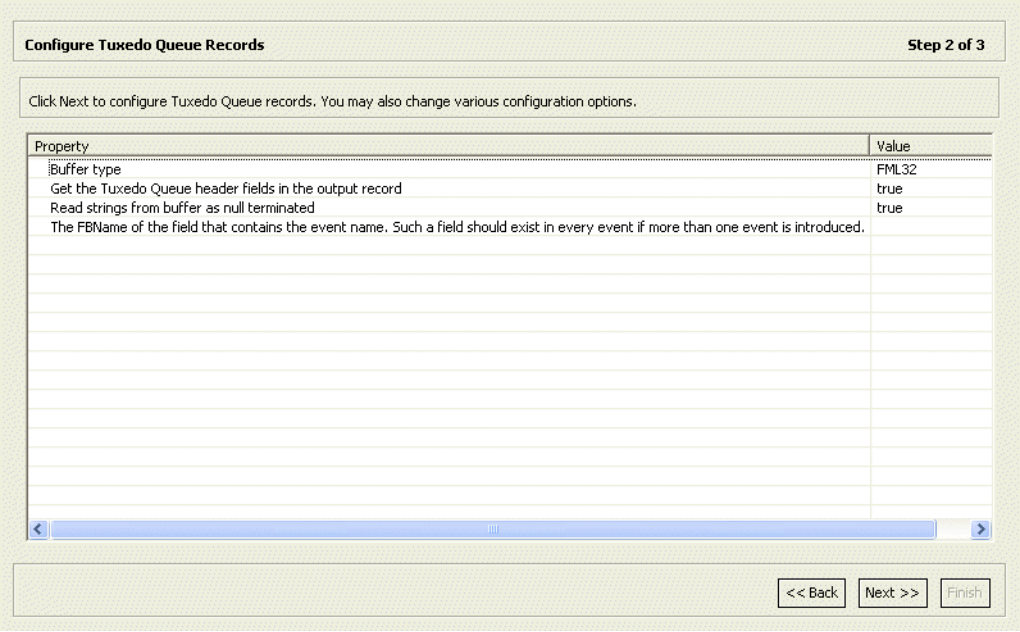
Figure 2–18 The Get Input Files screen



14. Click **Next**.

    The Configure Tuxedo Records screen is displayed, as shown in the following figure:

Figure 2–19 The Configure Tuxedo Queue Records screen



15. Ensure that the settings are correct for the following properties:

    - `Buffer type`: Indicates the buffer type, as read from the `FML/VIEW` file. This property should not be modified.
    - `Get Tuxedo Queue header field in the output record`: Indicates that the header filed of each record is read. The default setting is `true`.

- `Read strings from buffer as null terminated`: Indicates that strings are handled as null terminated. The default setting is `true`.

16. Specify the `FBName` of the field that contains the event name. This field is common to all incoming events and it should include the record name. This property is required only for `FML/FML32` files.
17. Click **Next** to generate the metadata definitions for the queue adapter.
18. Specify if you want to transfer the data to the server, then click **Finish**.

The Import wizard generates record structures, which are used for the record structures for the inbound interactions, and the metadata is imported based on the options specified and it is stored on the target platform.

An XML representation of the metadata is also generated. After performing the import, you can view the metadata in Oracle Studio Design perspective **Metadata** tab, under the **Imports** node of the Queue adapter. You can also make any fine adjustments to the metadata.

> **See Also:** Appendix A, "Advanced Tuning of the Metadata" for details about fine tuning the adapter metadata.

Defining the Tuxedo Queue Unstructured Records

You can manually define the required Tuxedo records when all the events in the Tuxedo queue are of the same type and are unstructured. Only one record is defined.

Perform the following steps to manually define the required Tuxedo records, as follows:

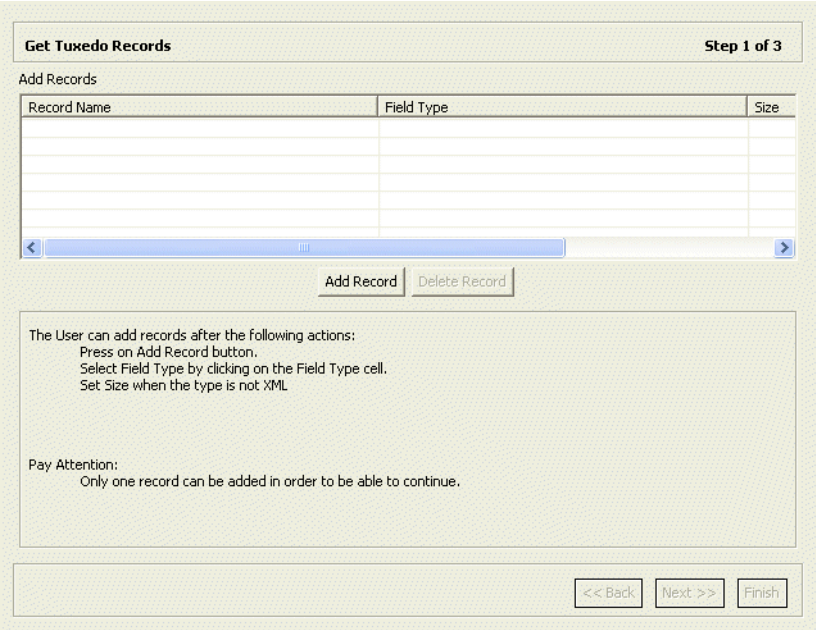
1. In the Configuration Explorer, select the Tuxedo Queue back-end adapter defined in Configuring the Tuxedo Queue Adapter on page 2-23.
2. Right-click the Tuxedo Queue adapter and select **Edit metadata**.

   The **Metadata** tab is displayed, with the queue adapter displayed under the Adapters list.
3. Right-click **Imports** and select **New Import**.

   The New Import screen is displayed.
4. Enter a name for the import. The name can contain letters, numbers and the underscore character only.
5. Select **Tuxedo Queue Import Manager for XML/STRING/CARRAY Buffers** from the **Import Type** list.
6. Click **Finish**.

   The Get Tuxedo Records screen is displayed, as shown in the following figure:

Figure 2–20 The Get Tuxedo Records screen



7. Click **Add Record**. A new record entry is added to the records list, with a default type.
8. Select the field type from the **Field Type** list. Your options are:
   - `STRING` (the default)
   - `CARRAY`
   - `XML`
   - `X_OCTET`
9. Specify the maximum buffer size in the Size column. This is not required if XML was selected as the field type.
10. Click **Next**.
11. Specify if you want to transfer the data to the target platform, then click **Finish**.

The record structure is generated, and the metadata is imported to and stored on the target platform.

An XML representation of the metadata is also generated. After performing the import, you can view the metadata in Oracle Studio Design perspective **Metadata** tab, under the **Imports** node of the Queue adapter. You can also make any fine adjustments to the metadata.

> **See Also:** Appendix A, "Advanced Tuning of the Metadata" for details about fine tuning the adapter metadata.

Viewing the XML Schema

The XML schema describing the adapter interactions can be viewed by selecting the **Source** tab of the adapter metadata properties in Oracle Studio Metadata explorer.

Creating XML Schemas

The XML schemas are created automatically during the import procedure, as described in "Generating Outbound Interactions" and in "Generating Inbound Interactions". XML schemas describe the adapter interactions and the input and output records for these interactions.

3

Integrating OracleAS Adapter for Tuxedo with OC4J

To deploy and integrate OracleAS Adapter for Tuxedo with Oracle Application Server Containers for J2EE (OC4J), you need to configure the J2CA 1.5 Tuxedo adapter.

This section includes the following topics:

Integrating OracleAS Adapters for Tuxedo with OC4J

Oracle Application Server provides a complete Java 2 Enterprise Edition (J2EE) environment that executes on the Java virtual machine (JVM) of the standard Java Development Kit (JDK). OC4J is J2EE certified and provides all J2EE specific containers, APIs, and services. OC4J supports the J2CA 1.5 standards.

J2CA defines standard Java interfaces for simplifying the integration of applications with the EIS. OracleAS adapters are deployed as resource adapters within the OC4J container.

The contract between the OC4J client application and the resource adapter is defined by the common client interface (CCI). The contract between the OC4J container and the resource adapter is defined by the service provider interface (SPI). The SPI API addresses the connection management, transaction management and the security management.

Connection management enables application components to connect to an EIS and leverage any connection pooling provided by the application server.

Transaction management enables an application server to use a transaction manager to manage transactions across multiple resource managers.

Lifecycle management contracts enable an application server to initialize a resource adapter instance during the deployment of the adapter or application server startup. In addition, it enables the application server to notify the resource adapter instance during server shutdown or undeployment of the adapter.

The lifecycle contract provides the mechanism for the application server to manage the lifecycle of the resource adapter instance.

Work management contracts enable the resource adapter to carry out its logic by using threads dispatched by an application server, rather than creating threads on its own. The handshake is done through a `Work` instance submission. This makes the

application server threads management more efficient, providing better control over their execution contexts (such as security and transaction).

Message inflow, which refers to inbound communications from an EIS to the application server (see "Configuring the J2CA 1.5 Tuxedo Queue Adapter for Inbound" on page 3-5).

> **See Also:** Oracle Application Server Adapter Concepts Guide, Oracle Application Server Containers for J2EE User's Guide, Oracle Application Server Containers for J2EE Services Guide, and Oracle Application Server Containers for J2EE Security Guide.

Integrating the J2CA 1.5 Tuxedo Adapter for Outbound

This section includes the following topics:

- Configuring the J2CA 1.5 Tuxedo Adapter for Outbound
- Using the CCI API to Develop Applications

Configuring the J2CA 1.5 Tuxedo Adapter for Outbound

To connect to the J2CA 1.5 Tuxedo adapter under Oracle Application Server, you need to set the relevant parameters in the connection factory. You can perform this task by using Oracle Enterprise Manager, or by carrying out the following steps:

1. Open the following file in an editor:

   root\j2ee\home\application-deployment\default\oracle\oc4j-ra.xml

   where *root* is the Oracle Application Server root directory.

2. In this file, set the following parameters for each connection.

```
<oc4j-connector-factories>
  <connector-factory location=" " connector-name="Oracle Legacy Adapter">
    <config-property name="userName" value=" "/>
    <config-property name="password" value=" "/>
    <config-property name="eisName" value=" "/>
    <config-property name="serverName" value=" "/>
    <config-property name="workspace" value=" "/>
    <config-property name="portNumber" value=" "/>
    <config-property name="persistentConnection" value=" "/>
    <config-property name="keepAlive" value=" "/>
    <config-property name="firewallProtocol" value=""/>
    <config-property name="connectTimeout" value=""/>
    <config-property name="encryptionProtocol" value=""/>
    <config-property name="encryptionKeyName" value=""/>
    <config-property name="encryptionKeyValue" value=""/>
    <config-property name="fakeXA" value=" "/>
    <config-property name="useNamespace" value=" "/>
    <config-property name="networkXMLProtocol" value=" "/>
    <config-property name="exposeEventStreamMetadata" value=" "/>
  </connector-factory>
</oc4j-connector-factories>
```

Note: `location` is the attribute that specifies the JNDI location where Oracle Application Server should bind the connection factory instance for application components.

Table 3–1 provides a detailed description of these parameters.

Table 3–1 OC4J Connection Properties for Outbound Interactions

Property	Description
`location`	Required. Specifies the JNDI location where Oracle Application Server should bind the connection factory instance for application components.
`eisName`	Required. Sets the name of the adapter to use. The adapter is defined in the Oracle Connect server using Oracle Studio, as described in "Configuring an Oracle Connect Adapter" on page 2-12.
`serverName`	Optional. Sets the TCP/IP address or host name where the Oracle Connect daemon is running. See Also: Appendix C, "Advanced Tuning of the Daemon" for details about the daemon.
`workspace`	Optional. Specifies the name of an Oracle Connect server workspace to use. The default workspace is `Navigator`. See Also: "Workspaces" for details about workspaces.
`portNumber`	Optional. Specifies the TCP/IP port where the Oracle Connect daemon is running on the server. The default port is `2551`.
`userName`	Optional. Specifies a user who can access the Oracle Connect server. The user is defined in the Oracle Connect daemon configuration. See also: "Daemon Security" and "WS Security" for details about users allowed to access an Oracle Connect server
`password`	Optional. Specifies a valid password for the user.
`persistentConnection`	Optional. Set to `true` or `false`. When set to `true`, connections can persist across multiple requests or connection context changes. It is recommended to set this property to `true`.
`keepAlive`	Optional. Set to `true` or `false`. When set to `true`, the socket used for the connection is always kept open. It is recommended to set this property to `true`.
`firewallProtocol`	Optional. Specifies the firewall protocol used: either `none` or `fixedNat` (the Nat protocol using a fixed address for the daemon). The default is set to `none`.
`connectTimeout`	Optional. Specifies the connection timeout in seconds. The default is 0, indicating that there is no connection timeout.
`encryptionProtocol`	Optional. Specifies the name of encryption protocol to use. The default is set to `null`. The RC4 protocol is supported.
`encryptionKeyName`	Optional. Specifies the name of the symmetric encryption key to use.

Table 3–1 (Cont.) OC4J Connection Properties for Outbound Interactions

Property	Description
`encryptionKeyValue`	Optional. Specifies the value of the symmetric encryption key to use.
`fakeXa`	Optional. When set to `true`, the XA APIs are internally converted to local transaction APIs. Do not set to `false`.
`useNamespace`	Optional. Set to `true` or `false`. When set to `true`, the metadata record schema uses namespaces.
`networkXMLProtocol`	Optional. Specifies how the XML is passed over the network. `Binary` or `Text` can be selected.
`exposeEventStreamMetadata`	Optional. When set to `true`, standard XSD schema is used for the Tuxedo/Q adapter inbound events metadata. When `false`, the native schema is used.

Configuring Multiple Adapters

Each J2CA 1.5 Tuxedo adapter, requires an entry in the `oc4j-ra.xml` file as described in "Configuring the J2CA 1.5 Tuxedo Adapter for Outbound" on page 3-2.

See Also: Oracle Application Server Adapter Concepts Guide

Updating Configuration Information

You can change the configuration settings for a resource adapter by editing the relevant `connector-factory` entry in the `oc4j-ra.xml` file. For these changes to take effect, you need to stop and restart Oracle Application Server.

Using the CCI API to Develop Applications

You can develop applications to run adapter interactions using the Common Client Interface (CCI) API.

Perform the following steps to use the CCI API with the J2CA 1.5 Tuxedo adapter:

1. Select a `ConnectionFactory` object for the J2CA 1.5 Tuxedo adapter.
2. Create a `Connection` object using the selected `ConnectionFactory`. A `Connection` is a handle to the underlying network connection to the EIS, which is identified in the `oc4j-ra.xml` file by the `serverName` property.
3. Create a `Connection` object using the selected `ConnectionFactory`. Specify the interaction properties using an `AttuInteractionSpec` object. The `AttuInteractionSpec` object has the following format:

   ```
   AttuInteractionSpec(java.lang.String name, int verb, int timeOut)
   ```

   The following table describes the properties that can be specified:

Table 3–2 Interaction-Spec Properties

Property	Description
`name`	Specifies the interaction name to be executed.
`verb`	Specifies the mode for the interaction: `SYNC_SEND`, `SYNC_SEND_RECEIVE`, or `SYNC_RECEIVE`.

Table 3–2 (Cont.) Interaction-Spec Properties

Property	Description
timeOut	Specifies the time (in milliseconds) to wait for an EIS to run the specified interaction.

The following is an InteractionSpec sample:

```
AttuInteractionSpec iSpeq = new AttuInteractionSpec("query",
javax.resource.cci.InteractionSpec.SYNC_RECEIVE, 60);
javax.resource.cci.RecordFactory rf = new AttuRecordFactory(con,
mcf.getLogger());
javax.resource.cci.MappedRecord queryRecord = rf.createMappedRecord("query"),
queryRecord.put("##text", "select * from disam:nation");
javax.resource.cci.Record oRec = interaction.execute(iSpec, queryRecord);
```

4. Invoke the `execute` method on the `interaction` to initiate a call to the EIS. Pass any data for the interaction as input and output records.
5. Once the interactions have been processed, close the `Interaction` and `Connection` objects.

Integrating the J2CA 1.5 Tuxedo Queue Adapter for Inbound

This section includes the following topics:

- Configuring the J2CA 1.5 Tuxedo Queue Adapter for Inbound
- Using the CCI API to Develop Message Endpoint Applications

Configuring the J2CA 1.5 Tuxedo Queue Adapter for Inbound

The provider of the endpoint must supply the following information in the endpoint deployment descriptor file, called `ejb-jar.xml`:

- The properties of the ActivationSpec class of the CICS Queue adapter
- The type of message listener implemented in the endpoint

The `orion-ejb.xml` file must provide the name of the resource adapter.

OC4J searches the deployment descriptor of the resource adapter for the message listener type and sets the properties that are defined in the endpoint deployment descriptor file for the respective ActivationSpec class.

The resource adapter supports message listener types and ActivationSpec classes as follows:

Table 3–3 Message Listener Types and their ActivationSpec Classes

Message Listener Type	ActivationSpec Class
javax.resource.cci.MessageListener	com.attunity.adapter.AttuActivationSpec
oracle.tip.adapter.api.OracleMessageListener	com.attunity.adapter.AttuOracleActivationSpec

Example 3–1 Endpoint Deployment Descriptor (ejb-jar.xml)

```
<ejb-jar>
  <display-name>AttuMDB1</display-name>
```

```
<enterprise-beans>
<message-driven>
    <display-name>Attu Test Receiver Bean</display-name>
    <ejb-name>AttuTestReceiverBean</ejb-name>
    <ejb-class>attutestreceiverbean.AttuTestReceiverBean</ejb-class>
    <messaging-type>javax.resource.cci.MessageListener</messaging-type>
    <transaction-type>Container</transaction-type>
    <activation-config>
      <activation-config-property>
        <activation-config-property-name>userName</activation-config-
            property-name>
      </activation-config-property>
      <activation-config-property>
        <activation-config-property-name>password</activation-config-
           property-name>
      </activation-config-property>
      <activation-config-property>
        <activation-config-property-name>firewallProtocol<
          /activation-config-property-name>
      </activation-config-property>
      <activation-config-property>
        <activation-config-property-name>connectTimeout</activation-
          config-property-name>
      </activation-config-property>
      <activation-config-property>
        <activation-config-property-name>encryptionProtocol
          </activation-config-property-name>
      </activation-config-property>
      <activation-config-property>
        <activation-config-property-name>encryptionKeyName
          </activation-config-property-name>
      </activation-config-property>
      <activation-config-property>
        <activation-config-property-name>encryptionKeyValue
          </activation-config-property-name>
      </activation-config-property>
      <activation-config-property>
        <activation-config-property-name>workspace
           </activation-config-property-name>
      </activation-config-property>
      <activation-config-property>
         <activation-config-property-name>portNumber
           </activation-config-property-name>
      </activation-config-property>
      <activation-config-property>
         <activation-config-property-name>useNamespace
           </activation-config-property-name>
      </activation-config-property>
      <activation-config-property>
         <activation-config-property-name>networkXMLProtocol
           </activation-config-property-name>
      </activation-config-property>
      <activation-config-property>
        <activation-config-property-name>idleTimeout
           </activation-config-property-name>
      </activation-config-property>
      <activation-config-property>
        <activation-config-property-name>messagesInBatch
         </activation-config-property-name>
        <activation-config-property-value>1</activation-config-property-value>
```

```
      </activation-config-property>
      <activation-config-property>
        <activation-config-property-name>support2PC
          </activation-config-property-name>
        <activation-config-property-value>false</activation-config-
           property-value>
      </activation-config-property>
      <activation-config-property>
        <activation-config-property-name>waitTime
          </activation-config-property-name>
        <activation-config-property-value>5</activation-config-property-value>
      </activation-config-property>
      <activation-config-property>
        <activation-config-property-name>retryInterval
         </activation-config-property-name>
        <activation-config-property-value>5
         </activation-config-property-value>
      </activation-config-property>
      <activation-config-property>
        <activation-config-property-name>eisName
        </activation-config-property-name>
        <activation-config-property-value>TUXGetEvents
        </activation-config-property-value>
      </activation-config-property>
      <activation-config-property>
        <activation-config-property-name>serverName
         </activation-config-property-name>
        <activation-config-property-value>drorr-xp
         </activation-config-property-value>
      </activation-config-property>
      <activation-config-property>
        <activation-config-property-name>exposeEventStreamMetadata
          </activation-config-property-name>
        <activation-config-property-value>true
          </activation-config-property-value>
      </activation-config-property>
    </activation-config>
    </message-driven>
  </enterprise-beans>
 <assembly-descriptor>
 <!-- NotSupported, Supports, Required, RequiresNew, Mandatory or Never  -->
   <container-transaction>
     <method>
       <ejb-name>
         AttuTestReceiverBean
       </ejb-name>
       <method-name>
         onMessage
       </method-name>
       <method-params>
       <method-param> javax.resource.cci.Record </method-param>
       </method-params>
     </method>
     <trans-attribute>
       NotSupported
     </trans-attribute>
   </container-transaction>
   </assembly-descriptor>
</ejb-jar>
```

The following table lists the properties that are relevant for inbound interactions.

Table 3–4 OC4J Connection Properties for Inbound Interactions

Property	Description
`location`	Required. Specifies the JNDI location where Oracle Application Server should bind the connection factory instance for application components.
`eisName`	Required. Sets the name of the adapter to use. The adapter is defined in the Oracle Connect server using Oracle Studio, as described in "Configuring an Oracle Connect Adapter" on page 2-12.
`messagesInBatch`	Specifies the maximum number of messages that can be moved to an endpoint in batch. The default is set to 1.
`support2PC`	Enables global transaction support, where applicable. The default is set to `False`. Do not set to `true`.
`waitTime`	Defines the maximum time (in seconds) for an empty transaction duration, and for the Tuxedo adapter to return a "no messages" response (which will cause the current transaction to close). The default is set to 30 seconds.
`retryInterval`	Defines the sleep time after any detected problem before restarting inbound activity. The default is set to 15 seconds.
`serverName`	Optional. Sets the TCP/IP address or host name where the Oracle Connect daemon is running. See Also: Appendix C, "Advanced Tuning of the Daemon" for details about the daemon.
`workspace`	Optional. Specifies the name of an Oracle Connect server workspace to use. The default workspace is `Navigator`. See Also: "Workspaces" for details about workspaces.
`portNumber`	Optional. Specifies the TCP/IP port where the Oracle Connect daemon is running on the server. The default port is `2551`.
`userName`	Optional. Specifies a user who can access the Oracle Connect server. The user is defined in the Oracle Connect daemon configuration. See also: "Daemon Security" and "WS Security" for details about users allowed to access an Oracle Connect server
`password`	Optional. Specifies a valid password for the user.
`firewallProtocol`	Optional. Specifies the firewall protocol used: either `none` or `fixedNat` (the Nat protocol using a fixed address for the daemon). The default is set to `none`.
`connectTimeout`	Optional. Specifies the connection timeout in seconds. The default is 0, indicating that there is no connection timeout.
`encryptionProtocol`	Optional. Specifies the name of encryption protocol to use. The default is set to `null`. The RC4 protocol is supported.

Table 3–4 (Cont.) OC4J Connection Properties for Inbound Interactions

Property	Description
encryptionKeyName	Optional. Specifies the name of the symmetric encryption key to use.
encryptionKeyValue	Optional. Specifies the value of the symmetric encryption key to use.
useNamespace	Optional. Set to true or false. When set to true, the metadata record schema uses namespaces.
networkXMLProtocol	Optional. Specifies how the XML is passed over the network. Binary or Text can be selected.
exposeEventStreamMetadata	When set to true (the default), EventStream schema is sent to the endpoint.

Using the CCI API to Develop Message Endpoint Applications

The endpoint uses the onMessage method. The following sample describes how the DOM is received from the CoreDomRecord class.

Example 3–2 onMessage method

```
public Record onMessage(Record inMessage)throws javax.resource.ResourceException {

    …
    CoreDOMWriter domW;
    domW = new CoreDOMWriter(false);
    Element outEl = ((CoreDomRecord)inMessage).getDom();
    String xml = domW.toXMLString(outEl);
    …

 return null;

}
```

The adapter ignores the return values.

If the Tuxedo Queue adapter describes two types of messages, employee and department, the XML data has the following input record structure:

- When the exposeEventStreamMetadata property is set to true:

```
<eventStream>
  < EMPLOYEE>
     ...
  </ EMPLOYEE>
  <DEPARTMENT>
     ...
  </DEPARTMENT>
</eventStream>
```

- When the exposeEventStreamMetadata property is set to false:

```
<getEventsResponse xmlns="noNamespace://QVREAD">
<event eventName="EMPLOYEE" timestamp="2005-08-23T15:23:18">
  < EMPLOYEE>
     ...
  </ EMPLOYEE>
</event>
<event eventName=" DEPARTMENT " timestamp="2005-08-23T15:23:18">
```

```
  <DEPARTMENT>
    ...
  </DEPARTMENT>
</event>
</getEventsResponse>
```

4

Integrating OracleAS Adapters for Tuxedo with Oracle BPEL Process Manager

To deploy and integrate OracleAS Adapters for Tuxedo with Oracle BPEL Process Manager, you need to configure BPEL Process Manager.

This section includes the following topics:

- Overview of Integrating OracleAS Adapters for Tuxedo with Oracle BPEL Process Manager
- Configuring Oracle BPEL Process Manager to interact with the OracleAS Adapter for Tuxedo

Overview of Integrating OracleAS Adapters for Tuxedo with Oracle BPEL Process Manager

Oracle BPEL Process Manager provides a comprehensive solution for creating, deploying, and managing BPEL business processes. Oracle BPEL Process Manager is based on the Service Oriented Architecture (SOA) to provide enterprises with flexibility, interoperability, reusability, extensibility, and rapid implementation of Web services and business processes. It reduces the overall costs of management, modification, extension, and redeployment of existing business processes. Each business activity is a self-contained, self-describing, and modular application whose interface is defined by the WSDL, and the business process is modeled as a Web Service.

A Web Service is first published and then composed or orchestrated into business flows. Publishing a service is implemented by taking a function within an existing application or system and making it available in a standard way, while orchestration is implemented by composing multiple services into an end-to-end business process. The interactions that are defined as part of the configuration of the OracleAS Adapter for Tuxedo are integrated into the orchestration as PartnerLinks. Every PartnerLink is linked to a WSDL that describes the Web service.

To integrating the OracleAS Adapter for Tuxedo with Oracle BPEL Process Manager, you must perform the following tasks in the specified order:

1. Installing and Configuring OracleAS Adapters for Tuxedo
2. Integrating OracleAS Adapter for Tuxedo with OC4J
3. Configuring Oracle BPEL Process Manager to interact with the OracleAS Adapter for Tuxedo

See Also: *Oracle Application Server Adapter Concepts Guide*.

Configuring Oracle BPEL Process Manager to interact with the OracleAS Adapter for Tuxedo

This section includes the following topics:

Setting up the Connection to the Oracle Connect Server

Perform the following steps to set up the connection to the Oracle Connect server:

1. Open the Oracle BPEL Admin window.
2. On the Server tab, on the Configuration tab, specify the following:
   - `LegacyServer`: The IP address of the server where Oracle Connect is installed. For a single server, the default is `localhost`.
   - `LegacyPort`: The port number of the server where Oracle Connect is installed. For a single port, the default is `2551`.
3. Repeat the previous step for each Oracle Connect server to be used by Oracle BPEL Process Manager. Use a comma as a separator between the different servers and ports.
4. Click **Apply**.
5. Restart the server where Oracle BPEL Process Manager is installed.

Checking Metadata Availability Using Oracle JDeveloper

Perform the following steps to verify that the metadata of the Oracle Connect server is available in Oracle BPEL Process Manager:

1. Open Oracle JDeveloper.
2. On the **Connections** tab, expand the **Integration Server** node to view the list of OC4J servers.
3. Expand the node of the OC4J server on which you configured the JCA 1.5 Tuxedo adapter (see Integrating OracleAS Adapter for Tuxedo with OC4J).
4. Under the **Adapters** node, expand the **Legacy** node to view a list of the Oracle Connect servers that you defined by using the Oracle BPEL Admin window.
5. Under the node of the Oracle Connect server whose metadata you want to check, expand the node of the daemon (**IRPCD**) to view a list of workspaces.

   A workspace includes adapters for either inbound or outbound applications.
6. Under the node of the workspace that contains the adapter that you want to work with, expand the node of the relevant adapter to view a list of interactions.

   If you selected a workspace that includes adapters for inbound applications, the adapter contains a single interaction only, called `EventStream`.
7. Double-click an interaction to view the WSDL.

Configuring the WSDL for Outbound Applications

When you build an outbound application, Oracle BPEL Process Manager automatically creates the WSDL that corresponds to the interaction. The WSDL specifies the name of the adapter's connection factory as the value of the `adapterInstanceJndi` attribute of the `<jca:address>` element in the `<service>` section. This name is generated automatically. You need to verify that a connection factory with this name exists on the OC4J server. If it does not, you need to create it, or change the name of the connection factory to the name of a connection factory that exists.

The following is an example of a WSDL for outbound applications:

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<definitions name="add" targetNamespace="http://xmlns.oracle.com/pcbpel/calc/add"
xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:legacyReq="noNamespace://calc"
xmlns:tns="http://xmlns.oracle.com/pcbpel/calc/add"
xmlns:plt="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
xmlns:jca="http://xmlns.oracle.com/pcbpel/wsdl/jca/"
xmlns:pc="http://xmlns.oracle.com/pcbpel/" xmlns:legacyRes="noNamespace://calc">
   <types>
       <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="noNamespace://calc" targetNamespace="noNamespace://calc"
elementFormDefault="qualified" attributeFormDefault="unqualified">
           <xsd:element name="binput" type="binput"/>
           <xsd:complexType name="binput">
               <xsd:attribute name="p1" type="xsd:int"/>
               <xsd:attribute name="p2" type="xsd:int"/>
           </xsd:complexType>
       </xsd:schema>
       <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="noNamespace://calc" targetNamespace="noNamespace://calc"
elementFormDefault="qualified" attributeFormDefault="unqualified">
            <xsd:element name="output" type="output"/>
           <xsd:complexType name="output">
               <xsd:attribute name="result" type="xsd:int"/>
           </xsd:complexType>
       </xsd:schema>
   </types>
   <message name="request">
       <part name="input_add" element="legacyReq:binput"/>
   </message>
   <message name="response">
       <part name="output_add" element="legacyReq:output"/>
   </message>
   <portType name="addPortType">
       <operation name="add">
           <input name="Input_add" message="tns:request"/>
           <output name="Output_add" message="tns:response"/>
       </operation>
   </portType>
   <binding name="addJCABinding" type="tns:addPortType">
       <jca:binding
XMLRecordConverterCallout="oracle.tip.adapter.fw.record.attunity.AttuXMLRecordConv
erterImpl"/>
       <operation name="add">
           <jca:operation FunctionName="add"
InteractionSpec="com.attunity.adapter.AttuInteractionSpec"
ExecutionTimeout="120"/>
           <input/>
           <output/>
```

```
        </operation>
    </binding>
    <service name="addService">
        <port name="addPort" binding="tns:addJCABinding">
            <jca:address adapterInstanceJndi="eis/legacy/calc"/>
        </port>
    </service>
    <plt:partnerLinkType name="addPartnerLinkType">
        <plt:role name="addRole">
            <plt:portType name="tns:addPortType"/>
        </plt:role>
    </plt:partnerLinkType>
/definitions>
```

Configuring the WSDL for Inbound Applications

When you build an outbound application, Oracle BPEL Process Manager automatically creates the WSDL that corresponds to the interaction, including the most important attributes of the `ActivationSpec` class, such as the name of the adapter, the server, the workspace, and the port number. The name of the workspace is the one that the `EventStream` interaction belongs to. All other properties have default values that you can modify.

The WSDL also specifies the name of the adapter's connection factory as the value of the `adapterInstanceJndi` attribute of the `<jca:address>` element in the `<service>` section. This name is generated automatically. If a connection factory with this name exists on the OC4J server, its values are taken. Otherwise, the properties specified by the `ActivationSpec` are used. If a value is specified by both the connection factory and the `ActivationSpec`, the `ActivationSpec` property overrides the value in the connection factory. If you want to use the value specified in the connection factory, you first need to delete the property from the `ActivationSpec`.

For a list of ActivationSpec properties, see the Oracle Application Server Containers for J2EE documentation.

The following is an example of a WSDL for outbound applications:

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<definitions name="eventStream"
targetNamespace="http://xmlns.oracle.com/pcbpel/testTuxedoQ1/eventStream"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:legacyReq="noNamespace://testTuxedoQ1"
xmlns:tns="http://xmlns.oracle.com/pcbpel/testTuxedoQ1/eventStream"
xmlns:plt="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
xmlns:jca="http://xmlns.oracle.com/pcbpel/wsdl/jca/"
xmlns:pc="http://xmlns.oracle.com/pcbpel/" xmlns:legacyRes="noNamespace://DEMO#">
   <types>
       <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="noNamespace://testTuxedoQ1" targetNamespace="noNamespace://testTuxedoQ1"
elementFormDefault="qualified" attributeFormDefault="unqualified">
           <xsd:element name="eventStream" type="eventStreamDescription"/>
           <xsd:complexType name="eventStreamDescription">
               <xsd:choice minOccurs="0" maxOccurs="unbounded">
                   <xsd:element name="EMPLOYEE_INFO__VAR_0__BIG_EMPLOYEE"
type="EMPLOYEE_INFO__VAR_0__BIG_EMPLOYEE"/>
                   <xsd:element name="EMPLOYEE_INFO__VAR_0__EMPLOYEE"
type="EMPLOYEE_INFO__VAR_0__EMPLOYEE"/>
               </xsd:choice>
           </xsd:complexType>
```

```
            <xsd:element name="EMPLOYEE_INFO__VAR_0__EMPLOYEE" type="EMPLOYEE_INFO_
_VAR_0__EMPLOYEE"/>
            <xsd:complexType name="EMPLOYEE_INFO__VAR_0__EMPLOYEE">
                <xsd:attribute name="LAST_NAME">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:maxLength value="15"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:attribute>
                <xsd:attribute name="FIRST_NAME">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:maxLength value="10"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:attribute>
                <xsd:attribute name="ADDRESS_1">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:maxLength value="15"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:attribute>
                <xsd:attribute name="ADDRESS_2">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:maxLength value="15"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:attribute>
                <xsd:attribute name="CARS" type="xsd:int"/>
                <xsd:attribute name="SALARY" type="xsd:int"/>
            </xsd:complexType>
            <xsd:element name="EMPLOYEE_INFO__VAR_0__BIG_EMPLOYEE" type="EMPLOYEE_
INFO__VAR_0__BIG_EMPLOYEE"/>
            <xsd:complexType name="EMPLOYEE_INFO__VAR_0__BIG_EMPLOYEE">
                <xsd:attribute name="ID">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:maxLength value="10"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:attribute>
                <xsd:attribute name="LAST_NAME2">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:maxLength value="100"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:attribute>
                <xsd:attribute name="FIRST_NAME2">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:maxLength value="100"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:attribute>
                <xsd:attribute name="ADDRESS_12">
                    <xsd:simpleType>
```

```
                        <xsd:restriction base="xsd:string">
                            <xsd:maxLength value="150"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:attribute>
                <xsd:attribute name="ADDRESS_22">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:maxLength value="150"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:attribute>
                <xsd:attribute name="CARS2" type="xsd:int"/>
                <xsd:attribute name="SALARY2" type="xsd:int"/>
                <xsd:attribute name="FILLER">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:maxLength value="1500"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:attribute>
            </xsd:complexType>
        </xsd:schema>
        <schema xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:tns="noNamespace://DEMO#" attributeFormDefault="qualified"
elementFormDefault="qualified" targetNamespace="noNamespace://DEMO#">
            <element name="eventStream">
                <complexType/>
            </element>
        </schema>
    </types>
    <message name="event">
        <part name="event_eventStream" element="legacyReq:eventStream"/>
    </message>
    <portType name="eventStreamPortType">
        <operation name="eventStream">
            <input name="Event_eventStream" message="tns:event"/>
        </operation>
    </portType>
    <binding name="eventStreamJCABinding" type="tns:eventStreamPortType">
        <pc:inbound_binding/>
        <operation name="eventStream">
            <jca:operation
ActivationSpec="com.attunity.adapter.AttuOracleActivationSpec"
EisName="testTuxedoQ1" ServerName="mvs5" UserName="" Password=""
Workspace="testTuxedoQ1" PortNumber="2551" FirewallProtocol="" ConnectTimeout="0"
EncryptionProtocol="" EncryptionKeyName="" EncryptionKeyValue=""
UseNamespace="true" NetworkXMLProtocol="" MessagesInBatch="50" Support2PC="false"
WaitTime="30" RetryInterval="15"/>
            <input/>
        </operation>
    </binding>
    <service name="eventStreamService">
        <port name="eventStreamPort" binding="tns:eventStreamJCABinding">
            <jca:address
ResourceAdapterClassName="com.attunity.adapter.AttuOracleResourceAdapter"
adapterInstanceJndi="eis/legacy/testTuxedoQ1"/>
        </port>
    </service>
    <plt:partnerLinkType name="eventStreamPartnerLinkType">
```

```
        <plt:role name="eventStreamRole">
            <plt:portType name="tns:eventStreamPortType"/>
        </plt:role>
    </plt:partnerLinkType>
/definitions>
```

5

Troubleshooting OracleAS Adapter for Tuxedo

Troubleshooting OracleAS Adapters for Tuxedo involves checking various definitions and properties in Oracle Connect, including daemon status, workspace options, server parameters, and various system logs.

This section contains the following topics:

Troubleshooting the Daemon

Troubleshooting the daemon and the communication between Oracle Application Server and OracleAS Adapters for Tuxedo is performed using Oracle Studio. It is used to monitor the daemon and server activity and control what happens to the daemon and server processes.

> **See Also:** Appendix C, "Advanced Tuning of the Daemon" for details about the configuration settings

This section contains the following:

Starting the Daemon

The daemon is started when OracleAS Adapters for Tuxedo are installed. In case you shut down the daemon, as described in "Shutting Down the Daemon" on page 5-2, you can restart the daemon as described in the following task.

Note: The daemon is started only on the platform where OracleAS Adapters for Tuxedo run. It cannot be started remotely using Oracle Studio.

Task: Starting the Daemon

- Enter the following command:

```
irpcd [-u username [-p password]] -l ip:2552 start
```

Where *username* is the name of a user with a permission to start the daemon, and *password* is the password for this user and *ip* is the ip address of the computer.

Shutting Down the Daemon

To shut down the daemon, use Oracle Studio, as follows.

1. From the **Start** menu, select **Programs**, **Oracle**, and then select **Studio**.
2. Select the computer defined in "Configuring OracleAS Adapters for Tuxedo in Oracle Studio" on page 2-8.
3. Right-click the computer and select **Open Runtime Perspective**.
4. In the Runtime Explorer, right-click the computer and select **Shutdown Daemon**.

Monitoring the Daemon During RunTime

Use the Runtime Manager perspective of Oracle Studio to monitor the daemon during runtime. Perform the following steps:

1. From the **Start** menu, select **Programs**, **Oracle**, and then select **Studio**.
2. Right-click the computer defined in "Configuring OracleAS Adapters for Tuxedo in Oracle Studio" on page 2-8 in the **Configuration Explorer** and select **Open Runtime Perspective**.

You can manage the daemon by expanding the relevant node, daemon, workspace or server process, and choosing the relevant option, as described in the following sections.

Daemon (Computer) Options

Right-click the daemon to display the options available for it, including the ability to display the daemon log.

The following table lists the available daemon options:

Table 5–1 Daemon Options

Option	Description
Edit Daemon Configuration	Opens the daemon editor to enable you to reconfigure the daemon. See Also: Appendix C, "Advanced Tuning of the Daemon" for details about the configuration settings.
Status	Checks the status of the daemon. The information about the daemon includes the name of the daemon configuration used, the active client sessions, and logging information.
Reload Configuration	Reloads the configuration after any changes. Any servers currently started are not affected by the changed configuration. See Also: Appendix C, "Advanced Tuning of the Daemon" for details about the configuration settings

Table 5–1 (Cont.) Daemon Options

Option	Description
View Log	Displays the daemon log. For details see "Daemon Logs" on page 5-4.
View Events	Displays the daemon events log.
Daemon Properties	Displays information about the computer where the daemon is running, such as the physical address and any username and password needed to access the computer
Shutdown Daemon	Shuts down the daemon on the computer.
Recycle servers	Closes all unused servers and prepares all active servers to close when the client disconnects. New connection requests are allocated with new servers.
Kill servers	Immediately closes all active and unused servers. **Note**: It is recommended to use this option with caution, as it may lead to data loss.
Rename	Enables to change the name of the daemon displayed in the Runtime Explorer.
Remove	Removes the computer from the Runtime Explorer.
Refresh	Refreshes the display.

Workspace Options

Right-click a workspace to display the options available for the workspace, including the ability to display the workspace log.

The following options are available at the workspace level:

Table 5–2 Workspace Options

Option	Description
Edit Workspace Configuration	Open the daemon editor to enable you to reconfigure the workspace. See Also: Appendix C, "Advanced Tuning of the Daemon" for details about the configuration settings.
Status	Checks the status of the workspace, whether it is available or not.
View Log	View the log for all servers for the workspace. For details see "Daemon Logs" on page 5-4.
View Events	Displays the workspace events log.
Recycle Servers	Closes all unused servers and prepares all active servers to close when the client disconnects. New connection requests are allocated with new servers.
Kill Servers	Immediately closes all active and unused servers. **Note**: It is recommended to use this option with caution, as it may lead to data loss.
Remove	Removes the selected workspace from the Runtime Explorer.
Disable	Disables the selected workspace.
Refresh	Refreshes the display.

Server Options

Right-click a server to display the options available for the server, including the ability to display the server log.

The options available at the server level are listed in the following tablel:

Table 5–3 Server Options

Option	Description
Status	Checks the status of the server. The information about the server includes the server mode and the number of active client sessions for the server.
View Log	View the server log. For details see "Daemon Logs" on page 5-4.
View Events	Displays the server events log.
Kill server	Ends the server process, regardless of its activity status. **Note**: It is recommended to use this option with caution, as it may lead to data loss.
Refresh	Refreshes the display.

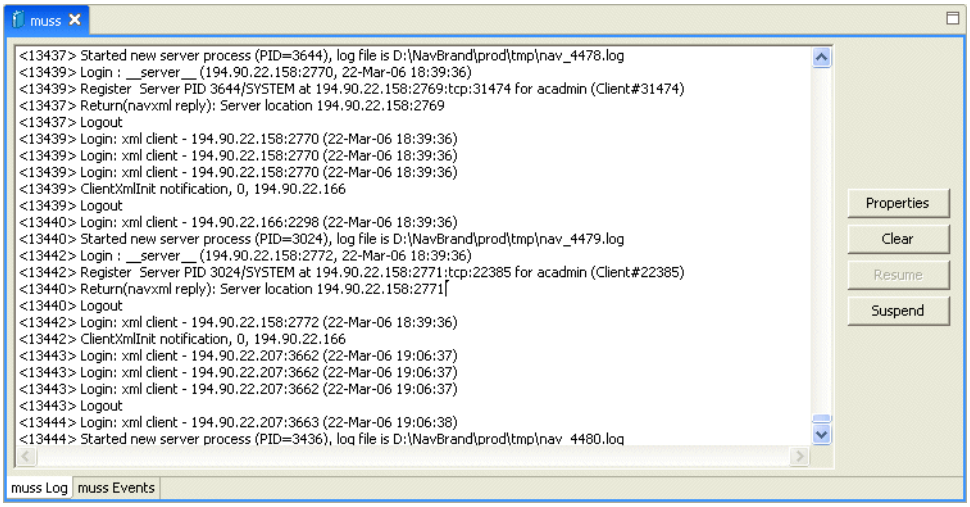
Daemon Logs

Oracle Connect produces a number of logs that you can use to troubleshoot problems. The daemon manages the following logs:

- The Daemon log
- The Workspace log
- The Server process log

The Runtime Manager perspective of Oracle Studio provides a monitor for these logs, as shown in the following figure:

Figure 5–1 The Legacy System Log tab



Display the required log by right-clicking the level item want (daemon, workspace or server) and selecting **View Log**. Each log is displayed in a separate tab. You can flick between logs by clicking the appropriate tab.

The Daemon Log Monitor

The daemon log displays activity between clients and the daemon, including clients logging in and logging out from the daemon.

You can change the level of logging by clicking Properties. The following levels of logging are available:

- `none`: The log displays who has logged in and out from the daemon.
- `error`: The log displays who has logged in and out from the daemon and any errors that have been generated.
- `debug`: The log displays who has logged in and out from the daemon, any errors that have been generated, and any tracing that has been specified in the daemon configuration.

See Also: "Daemon Logging" on page C-3.

The Workspace Log Monitor

The workspace log displays information about the workspace being used by the client.

You can change the level of logging by clicking Properties. The following levels of logging are available:

- `none`: The log displays who has connected and disconnected from the server process.
- `error`: The log displays who has connected and disconnected from the server process and any errors that have been generated.
- `debug`: The log displays who has connected and disconnected from the server process, any errors that have been generated, and any tracing that has been specified in the daemon configuration.

See Also: "WS Logging" on page C-13.

The Server Log Monitor

The server log displays activity between clients and the server process used by that client to handle the client request.

You can change the level of logging by clicking Properties. The following levels of logging are available:

- `none`: The log displays who has connected and disconnected from the server process.
- `error`: The log displays who has connected and disconnected from the server process and any errors that have been generated.
- `debug`: The log displays who has connected and disconnected from the server process, any errors that have been generated, and any tracing that has been specified in the daemon configuration.

See Also: "WS Logging" on page C-13.

Resolving Communication Errors

When Oracle Studio disconnects from the Tuxedo computer, the computer is displayed in Oracle Studio with an `X` in a red circle. If this situation occurs, try to access the computer later.

The following table describes the various scenarios that may exist when Oracle Application Server disconnects from the platform where OracleAS Adapters for Tuxedo run.

Table 5–4 Scenarios When a Client Is Disconnected

Scenario	Idle (Not Processing a Client Request)	Processing a Client Request
Explicit Disconnect (client explicitly closes connection or client program terminates)	The server is immediately notified of the disconnect and either becomes available for use by another client or terminates (if it is not reusable).	The server does not know that the client has disconnected and continues processing. When processing completes, the server tries to reply to the client and immediately gets an error that the connection was lost. The server either becomes available for use by another client or terminates (if it is not reusable).
Abrupt Disconnect (client closed without proper shutdown or client system hanged and communication disconnected)	The server does not know that the client has disconnected and remains in the idle state. After timing out based on whichever comes first of the value for the client idle timeout daemon workspace parameter or the TCP/IP KEEPALIVE parameter, the server is notified of the disconnect and either becomes available for use by another client or terminates (if it is not reusable).	The server does not know that the client has disconnected and continues processing. When processing completes, the server tries to reply to the client. After an interval (typically several minutes, depending on the TCP/IP configuration), during which the TCP/IP subsystem retries sending the message to the client, the server assumes that the client has terminated and notifies the server that the connection has been closed. The server either becomes available for use by another client or terminates (if it is not reusable).

To troubleshoot client/server communication problems, you need to be familiar with the following:

- Daemon configuration settings.
- Oracle Connect security.
- TCP/IP subsystem. OracleAS Adapters for Tuxedo uses TPC/IP for internal intercomputer communications.
- System details, such as the account name and password of the administrator account, the IP address of the computers involved and whether a portmapper is being used.

Resolving Specific Errors

The following error messages relate to errors received from Oracle Connect.

C005: Could not open the IRPCD log file for write.

Cause: The daemon was not able to create or write to its log file. The log file is viewed using the Oracle Studio Runtime perspective, as described in "Daemon Logs" on page 5-4. The log file location is set in the daemon configuration as described in "Daemon Logging" on page C-3.

Action: Check that the account where the daemon runs has permission to generate/write to the log file.

Action: Check the path specified for the log file in the daemon configuration.

Action: Check that there is no existing log file owned by another user at the specified location.

Action: Ensure that the disk device is not full.

C007: Server initialization failed.

Cause: The daemon failed to start its network service.

Action: Check the processes being run on the system to see whether another daemon or program is using the port specified in the `oc4j-ra.xml` file for the adapter.

Action: Check the TCP/IP subsystem on the current computer by trying to ping it or run FTP or Telnet to or from it.

Action: Check whether the daemon has privileges to use the TCP/IP services on the current computer with the port specified in the oc4j-ra.xml file for the adapter.

C008: Setting server event handler failed.

Cause: Internal error.

Action: Contact Oracle Support Services.

C009: IRPCD process has been terminated by user request.

Cause: This message is informational only. The daemon successfully shut down.

Action: No action required.

C00A: Application %s not found.

Cause: The requested workspace does not exist.

Action: Check that the workspace defined in the oc4j-ra-xml file is also defined in the daemon configuration on the platform where OracleAS Adapter for Tuxedo runs. Use the Status option in the Runtime Manager perspective.

C00B: Invalid IRPCD client context.

Cause: A non-Oracle Connect program is trying to connect to the daemon.

Action: Check the processes and terminate the relevant process with a system command.

C00C: Daemon request requires a server login.

Cause: A non-Oracle Connect server or program was trying to use a daemon service reserved for Oracle Connect servers.

Action: Check the processes and terminate the relevant process with a system command.

C00D: Daemon request requires a client login.

Cause: The requested daemon requires a valid client login, which was not supplied.

Action: Reissue the command and specify a username and password.

Action: Edit the user profile in Oracle Studio to specify a valid username and password for the platform where OracleAS Adapters for Tuxedo run.

> **See Also:** "Configuring Run-Time User Access to OracleAS Adapter for Tuxedo" on page 2-11.

C00E: Daemon request requires an administrator login.

Cause: The requested daemon service requires an administrative login.

Action: Edit the daemon security in Oracle Studio to specify a valid administrator username and password.

See Also: "Daemon Security" on page C-5.

C00F: Anonymous client logins are not allowed.

Cause: The daemon is configured to require a valid username and password, which were not supplied.

Action: Enable anonymous client access in daemon security in Oracle Studio.

See Also: "Daemon Security" on page C-5.

Action: Edit the user profile in Oracle Studio to specify a valid username and password for the platform where OracleAS Adapters for Tuxedo runs.

See Also: "Configuring Run-Time User Access to OracleAS Adapter for Tuxedo" on page 2-11.

C010: Anonymous server logins are not allowed.

Cause: Internal error.

Action: Contact Oracle Support Services.

C011: Client has already timed out.

Cause: A server process was started on behalf of a client and the client has timed out before the server completed its startup.

Action: Increase the Connect timeout value for the server workspace in the WS Info. section of the daemon configuration.

See Also: "WS Info." on page C-7.

C012: Invalid username/password.

Cause: Invalid username/password supplied when logging on to the daemon.

Action: See the daemon log file for the reason that the username/password were not accepted.

Action: Edit the user profile in Oracle Studio to specify a valid username and password for the platform where OracleAS Adapters for Tuxedo run.

See Also: "Configuring Run-Time User Access to OracleAS Adapter for Tuxedo" on page 2-11.

C014: Client connection limit reached - try later.

Cause: The maximum number of server processes for the workspace has been reached, and none of the active servers could accept the client connection.

Action: Increase the value of the `Set maximum number of servers` and `Maximum` parameter for the `Clients per server limit` field in the **WS Server** tab of the daemon configuration.

See Also: "WS Server" on page C-9.

Action: Try running the command later.

C015: Failed to start server process.

Cause: The Oracle Connect daemon failed to start a server process or the started server failed upon starting up.

Action: See the daemon and server log files for the reason the server did not start. For example, you might receive an message with a reason specified in the log file similar to the following: `[C015] Failed to start NAVIGATOR server process: No server account name defined for anonymous client; code: -1601: SQL code: 0`

Action: If you use impersonation, check the user profile on the client. Also see C069.

C016: Unexpected server state.

Cause: Internal error.

Action: Contact Oracle Support Services.

C017: Active daemon clients exist. Shutdown canceled.

Cause: One or more clients are still connected to the daemon.

Action: Wait until all the clients log off the daemon and then retry the shutdown operation.

C019: Request is not granted because someone else is locking it.

Cause: A request to lock a resource managed by the daemon was denied because another user has locked the resource.

Action: Wait for the other user to release the resource.

C01A: Lock %s not found.

Cause: A request to free a resource was denied because the caller did not lock that resource (for example, another user shut down the daemon you are working with).

Action: Contact Oracle Support Services.

C01B: Unexpected error in %s.

Cause: Internal error.

Action: Contact Oracle Support Services.

C01C: Cannot update configuration without _APPLICATIONS lock.

Cause: Internal error.

Action: Contact Oracle Support Services.

C01D: Need to lock the application first.

Cause: Internal error.

Action: Contact Oracle Support Services.

C01F: Cannot set configuration of a deleted application.

Cause: Internal error.

Action: Contact Oracle Support Services.

C020: Failed in looking up host name (gethostname())

Cause: Cannot connect to the remote computer.

Action: Check that the name specified for the computer in the oc4j-ra-xml file is correct.

Action: Check that a domain name server (DNS) is available to look up the host name.

Action: Check the TCP/IP subsystem on the computer by trying to ping it or run ftp or telnet to or from it.

C021: Required variable %s not found

Cause: An environment variable required by the Oracle Connect server was not defined when the server started up.

Action: Check whether the startup script makes any changes to the environment variables used by Oracle Connect.

Action: Check whether the system-defined environment size is sufficiently large for Oracle Connect.

C022: Server failed to connect and register with the daemon.

Cause: An Oracle Connect server started by the daemon was not able to connect or register back with the daemon.

Action: Try to connect again.

Action: Increase the Connect timeout value for the server workspace in the WS Info. section of the daemon configuration.

See Also: "WS Info." on page C-7.

Action: Check that the startup script for the workspace launches the correct version of Oracle Connect.

Action: Increase the value of the `Set maximum number of servers` and `Maximum` parameter for the `Clients per server limit` in the WS Server section of the daemon configuration.

See Also: "WS Server" on page C-9.

C023: Call made to unregistered module %d.

Cause: Internal error.

Action: Contact Oracle Support Services.

C024: Failed to create a socket.

Cause: An error occurred within the TCP/IP subsystem.

Action: Check whether you have sufficient system privileges.

Action: Check the TCP/IP subsystem on the computer by trying to ping it or run ftp or telnet to or from it.

C025: Failed to set socket option %s

Cause: An error occurred within the TCP/IP subsystem.

Action: Check whether you have sufficient system privileges.

Action: Check the TCP/IP subsystem on the computer by trying to ping it or run ftp or telnet to or from it.

C026: Failed to bind server to port %s

Cause: An Oracle Connect server or daemon was not able to bind to the specified port.

Action: Check whether another program is holding the port that was specified in the oc4j-ra-xml file for the adapter.

Action: Check whether you have sufficient system privileges.

C027: Cannot create TCP service for %s

Cause: An error occurred within the TCP/IP subsystem

Action: Check the TCP/IP subsystem on the computer by trying to ping it or run ftp or telnet to or from it.

C028: Unable to register (%s, %d, tcp)

Cause: This error may happen when a portmapper is used (*host*:a) but the portmapper is not available.

Action: Enable the portmapper.

Action: Avoid using the portmapper (by not using ":a" when starting the daemon).

C029: Failed to create a server thread

Cause: Internal error.

Action: Contact Oracle Support Services.

C02A: Server thread failed to start

Cause: Internal error.

Action: Contact Oracle Support Services.

C02B: Stopping the %s server - no client

Cause: A server that was started by the Oracle Connect daemon to service a client did not get a client connection request within one minute. The server terminates.

Action: In most cases, the client was terminated by a user request, so no specific action is required.

Action: If no client can connect to the server, it may be that the server has multiple network cards and the Oracle Connect daemon is not aware of this. In this case, start the daemon with an IP address.

C02C: Unexpected event - a termination signal intercepted

Cause: Internal error.

Action: Contact Oracle Support Services.

C02D: Modified transport, context unknown/lost

Cause: Internal error.

Action: Contact Oracle Support Services.

C02F: Corrupted arguments passed to procedure

Cause: Internal error.

Action: Contact Oracle Support Services.

C030: Unable to free arguments for %s() of %s

Cause: Internal error.

Action: Contact Oracle Support Services.

C031: Cannot register a non-module RPC %s

Cause: Internal error.

Action: Contact Oracle Support Services.

C032: An IRPCD program is required

Cause: Internal error.

Action: Contact Oracle Support Services.

C033: An IRPCD super-server is required for module events

Cause: Internal error.

Action: Contact Oracle Support Services.

C034: An invalid super-server module ID was specified, %d

Cause: Internal error.

Action: Contact Oracle Support Services.

C035: Out of memory

Cause: Not enough memory to service a client request.

Action: Increase process memory quota or add memory to the system.

C036: Failed to register RPC procedure module %s

Cause: Internal error.

Action: Contact Oracle Support Services.

C037: Failed to register an invalid RPC procedure number %x

Cause: Internal error.

Action: Contact Oracle Support Services.

C038: Cannot re-register RPC procedure number %x

Cause: Internal error.

Action: Contact Oracle Support Services.

C042: Remote call to %s failed; %s

Cause: Remote call to API failed.

Action: Check the daemon log file.

Action: If necessary, change the level of detail written to the log file to help resolve the problem.

See Also: "Daemon Logging" on page C-3.

C043: Failed to connect to host %s;%s

Cause: The remote host is not correctly defined to Oracle Connect or is not working.

Action: Check the remote computer definition in the oc4j-ra.xml file for the adapter.

Action: Check that the daemon is up on the platform where OracleAS Adapters for Tuxedo run. Use the Status option in the Runtime Manager perspective.

Action: Check the network connection by trying to ping the host computer or run ftp or telnet to or from it.

C045: Failed to create a service thread

Cause: The server failed to create a thread to service a client request.

Action: A system or process quota limit has been exceeded. Either increase the quota or lower the `Clients per server limit` field value in the WS Info. section of the daemon configuration.

See Also: "WS Info." on page C-7.

C047: %s out of memory

Cause: Not enough memory was available to Oracle Connect to complete a requested operation.

Action: Terminate unnecessary processes running on the server.

Action: Add more memory to the system.

Action: Allow the process to use more memory.

Action: Limit the number of processes the daemon may start. If the demand for servers exceeds the number of available servers, clients get a message telling them the maximum number of servers has been reached and asking them to try again later.

C066: Communication error with the server%s

Cause: Connection to the Oracle Connect daemon or server failed, or an established session with a server has failed.

Action: Check the remote computer definition in the oc4j-ra-xml file.

Action: Check that the daemon is up on the platform where OracleAS Adapters for Tuxedo run. Use the Status option in the Runtime Manager perspective.

Action: In case of a network problem, check the network connection by trying to ping the host computer or run ftp or telnet to or from it.

C067: Unexpected error occurred in server function %s

Cause: One of the server functions has exited with an exception, such as an abend, or an Invalid Instruction.

Action: Contact Oracle Support Services.

C068: Fail to login daemon

Cause: The daemon is not running on the server computer.

Action: Use the Status in Oracle Studio Runtime Manager perspective to check whether a daemon is running on the server

Action: Have the system administrator reinstall Oracle Connect on the server.

C069: Fail to get server

Cause: The Oracle Connect daemon on the server computer could not start a server process to serve the client. A separate message provides more detail on why the server process could not start.

Action: There are many possible causes of this error. If the cause is not clear from the related message, see the Oracle Connect daemon log file on the server

Action: The resolution to this error is highly dependent on the particular cause. The following are some typical causes and resolutions.

Action: Some process creation quota was exceeded. Either try again later or increase the quota or the other relevant system resources.

Action: The server startup script failed.

Action: The username given is not allowed to use the requested server. Use an authorized username.

Action: A limit on concurrent clients for a server has been reached. Try again later.

Action: If you use impersonation, check the user profile on the client. Also see C015.

C06A: Failed to connect to server

Cause: The server assigned to the client did not accept the client connection. A separate message provides more detail about why the server process did not accept the connection.

Action: See the daemon and server log files for the reason that the server was not available to accept its assigned client.

Action: If a multithreaded server is used and many clients are trying to connect to it at the same time, some may get a Connection Refused error if the TCP/IP request queue fills up.

C06B: Disconnecting from server

Cause: A network failure, or a server computer failure or a server program failure caused the connection to stop. The currently active transaction is stopped as well.

Action: Oracle Connect automatically tries to reestablish a connection with a server upon the next SQL command issued against the server. Once the network or computer failure is corrected, the connection to the daemon is reestablished automatically.

C070: Server failed to send reply to the client

Cause: Server terminated unexpectedly.

Action: Unless the client was intentionally stopped (for example, using Control-C), contact Oracle Support Services.

C071: Connection to server %s was disconnected. Cursors state was lost.

Cause: Either a network failure, a server computer failure, or server program failure caused the connection to stop. The currently active transaction is stopped as well.

Action: Normally, Oracle Connect automatically tries to create a new session with the server upon the next attempt to access the server. If the network and server are accessible, the next operation should succeed. Otherwise, the network or server should be fixed before connection can be resumed.

Action: In case of a server malfunction that is not related to callable user code, contact Oracle Support Services.

C072: Reconnect to server %s

Cause: This is an informational message only. The client has reestablished its connection with the server.

Action: No action is required.

C073: The parameters passed to the admin server are invalid: %s

Cause: Internal error.

Action: Contact Oracle Support Services.

C074: No authorization to perform the requested operation (%s)

Cause: User/account has insufficient privileges.

Action: Grant administrative privileges to the user/account using the Administrator parameter of the Daemon Security or WS Security sections in the daemon configuration.

> **See Also:** "Daemon Security" on page C-5 or "WS Security" on page C-16.

C075: Failed to register daemon in the TCP/IP service table

Cause: Registration of the daemon in the TCP/IP services file has failed.

Action: Check that the account running the daemon has the permissions to update the TCP/IP services file.

E001: Failed in lock/release operation

Cause: A lock or release operation of a global resource has failed. A separate message provides more details. The separate message specifies the cause of this error.

Action: There are various causes for this error, including lack of sufficient privileges or a system resource shortage.

J0006: Operation on already closed connection was requested

Cause: A request using a connection that was closed was attempted.

Action: Reopen the connection and try again.

J0028: Internal Error: Unknown XML tag %s

Cause: Internal error.

Action: Contact Oracle Support Services.

J0030: Internal Error: Method %s needs to be overwritten

Cause: Internal error.

Action: Contact Oracle Support Services.

J0031: Internal Error: Required attribute %s not found in %s verb

Cause: Internal error.

Action: Contact Oracle Support Services.

J0032: Internal Error: %s ACP object was returned instead of %s as expected

Cause: Internal error.

Action: Contact Oracle Support Services.

J0033: Internal Error: Attempt to work with closed socket

Cause: Internal error.

Action: Contact Oracle Support Services.

J0034: Internal Error: corrupted message; %s bytes read instead of %s as expected

Cause: XML sent from the client to the server has become corrupted.

Action: Check compression settings for XML transferred from the client to the server. If the settings are correct, retry sending the request from the client to the server.

J0035: Internal Error: Invalid redirection address %s returned by daemon

Cause: Internal error.

Action: Contact Oracle Support Services.

J0036: %s: %s

Cause: One of the following errors was received from the server: 0 - server.internalError, 1 - client.xmlError, 2 - client.requestError, 3 - client.noActiveConnection, 4 - server.resourceLimit, 5 - server.redirect, 6 - client.noSuchResource, 7 - client.authenticationError, 8 - client.noSuchInteraction, 9 - client.noSuchConnection, 10 - server.notImplemented, 11 - server.xaProtocolError, 12 - server.xaUnknownXID, 13 - server.xaDuplicateXID, 14 - server.xaInvalidArgument, 15 - client.autogenRejected, 16 - server.xaTransactionTooFresh, 17 - server.resourceNotAvailable, 18 - client.authorizationError, 19 - server.configurationError

Action: Review the server log file to determine the problem.

J0037: Internal Error: No ACP response when %s was expected

Cause: Internal error.

Action: Contact Oracle Support Services.

J0039: Internal Error: ACP root is not found in the XML

Cause: Internal error.

Action: Contact Oracle Support Services.

J0040: Internal Error: Input record is required for interaction %s execution

Cause: Internal error.

Action: Contact Oracle Support Services.

J0048: Invalid metadata type %s is passed to %s function

Cause: A request for metadata was not fulfilled.

Action: Check the validity of the request.

J0050: Key of the put method must be of type string

Cause: In either a GET or PUT operation, the key must be a string.

Action: Change the key used in the operation to a valid key.

J0059: Value %s is invalid for attribute %s

Cause: A request for metadata was not fulfilled.

Action: Check the validity of the request.

J0068: Value must be of type string

Cause: In a PUT operation, the value must be a string.

Action: Change the value used in the operation to a valid value.

J0069: Value must be of type MappedRecord

Cause: In a PUT operation, the value must be a mapped record.

Action: Change the value used in the operation to a valid value.

J0070: Value must be of type "MappedRecord[]"

Cause: In a PUT operation, the value must be mapped record array.

Action: Change the value used in the operation to a valid value.

J0071: Bad key for mapped record, #element or #element[] is required

Cause: In a PUT operation, the value must be mapped record array.

Action: Change the key used in the record to a valid key.

J0072: Value must be of type Object[]

Cause: In a PUT operation, the value must be mapped record array.

Action: Change the value used in the operation to a valid value.

J0078: In nonpersistent connection and non keep alive encryption is not supported - ignored

Cause: Encryption is not supported for nonpersistent connections.

Action: There is no action to take. This warning can be ignored.

J0079: Invalid argument passed to %s - Argument: %s, Value: %s

Cause: The value pass.

Action: Change the argument used to a number.

6

Advanced Features of OracleAS Adapter for Tuxedo

Oracle Connect includes a number of tuning parameters that can improve performance. Specifically, the daemon can be configured to optimize communication between the OracleAS Adapter for Tuxedo and a client. In addition, the binding environment can be tuned to optimize the request handling.

This section contains the following topics:

Configuring the Daemon for High Availability

The daemon workspace is responsible for allocating server processes to clients. You can configure a workspace to use a pool of server processes so that a server process is always available for a client request. Use Oracle Studio to maintain daemon and daemon workspace parameters to control the allocation of server processes and their management in a pool.

The daemon manages a pool of server processes on the legacy platform. Oracle Application Server manages connection pooling of the J2CA 1.5 Tuxedo adapter.

You can also have a number of daemon workspace configurations, enabling you to create individual workspaces for use with different adapters.

Adding a New Daemon Workspace Configuration

Use Oracle Studio to add a new daemon configuration. You can set up different daemon configurations for different situations. On the computer where Oracle Connect is installed, perform the following steps:

1. From the **Start** menu, select **Programs, Oracle**, and then select **Studio**.
2. In the Configuration Explorer, click the node of the computer defined in "Configuring OracleAS Adapters for Tuxedo in Oracle Studio" on page 2-8.
3. Expand the **Daemons** node. The available daemon configurations are listed.
4. Right-click **IRPCD** and select New Workspace.

5. In the New daemon workspace screen, specify a name for the new workspace and optionally, provide a description.
6. Specify whether you want default settings or copy the properties of an existing workspace.

   To copy the properties of an existing workspace, click **Ellipsis** and select the workspace from which you want to copy the properties.
7. Click **Next**. The Select Scenario screen is displayed.
8. Select **Application Server** using connection pooling and click **Next**.
9. Continue through the wizard, specifying the required values for the workspace.
10. To complete the workspace definition, click **Finish**.

The new workspace is now displayed under the IRPCD daemon node.

Editing the Workspace

You edit a workspace by using the tabs described in the following table:

Table 6–1 Workspace Definition tabs

Tab	Description
WS Info	Specifies general information including the server type, the command procedure used to start the workspace, the binding configuration associated with this workspace and the timeout parameters.
WS Server Mode	Specifies workspace server information including features that control the operation of the servers started up by the workspace and allocated to clients.
WS Logging	Specifies parameters for logging and the format to use for a log file.
WS Security	Specifies administration privileges, user access, ports available to access the workspace and workspace account specifications.
WS Governing	This tab is not applicable for use with OracleAS Adapter for Tuxedo.

To access these tabs, perform the following steps using Oracle Studio:

1. From the **Start** menu, select **Programs, Oracle**, and then select **Studio**.
2. In the Configuration Explorer, click the node next to the computer defined in "Configuring OracleAS Adapters for Tuxedo in Oracle Studio" on page 2-8.
3. Expand the **Daemons** node. The daemon configurations available on this computer are listed.
4. Expand the **IRPCD node**. The daemon workspaces are listed.
5. Right-click the required workspace and select Edit Workspace.
6. Click the required tab, which contains the information you want to edit.

> **Note:** For full details of the tabs and the fields in these tabs, refer to "Workspaces" on page C-7.

7. After editing the workspace, click **Save**.

Configuring the Server Mode

The server mode determines how the daemon starts up new processes. The daemon supports the following server modes:

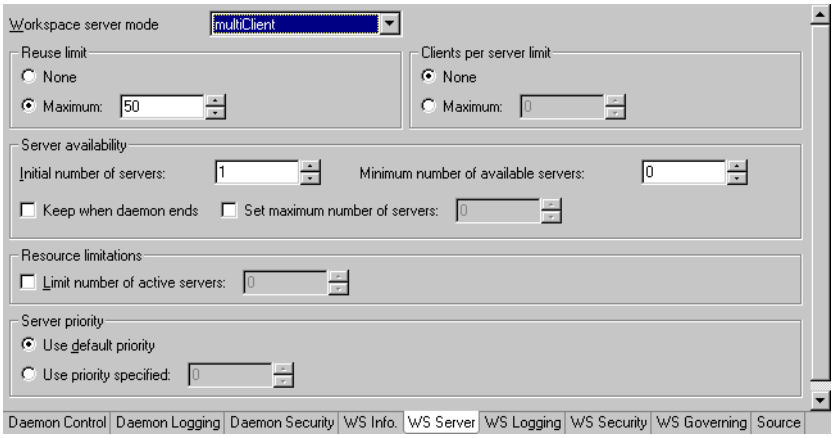
- `singleClient`: Each client receives a dedicated server process. The account in which a server process runs is determined either by the client login information or by the specific server workspace.
- This mode enables servers to run under a particular user account and isolates clients from each other (because each receives its own process). However, this server mode incurs a high overhead due to process startup times and may use a lot of server resources (because it requires as many server processes as concurrent clients).
- `multiClient`: Clients share a server process and are processed serially.
- This mode has low overhead because the server processes are already initialized. However, because clients share the same process, they may impact one another, especially if they issue lengthy queries.

  The number of clients that share a process is determined by the Clients per server limit (the maximum number of concurrent clients a server process for the current workspace accepts).
- `multiThreaded`: When Tuxedo runs on a Windows platform, the server process can be multi-threaded.
- `reusable`: This is an extension of the single client mode. Once the client processing finishes, the server process does not die and can be used by another client, reducing startup times and application startup overhead.
- This mode does not have the high overhead of single client mode because the servers are already initialized. However, this server mode may use a lot of server resources (because it requires as many server processes as concurrent clients).

  The other modes can be set so that the server processes are reusable by setting the number of times a process can be reused with the Reuse limit value (the maximum number of times a particular server process can be reused or how many clients it can serve before it is retired). Reuse of servers enhances performance because it eliminates the need to repeat initializations. However, reuse runs a risk of higher memory leakage over time. The default for the Reuse limit field value is None, indicating that no reuse limit is enforced.

Set the server mode in the **WS Server** tab of the daemon workspace editor, as shown in the following figure:

Figure 6–1 The WS Server tab



When using any of the server modes you can specify a pool of server processes. These server processes are started when the daemon starts and are maintained in a pool. The server processes are available for use by new client requests from the pool, saving initialization time. Instead of starting a new server process each time one is requested by a client, the client receives a process immediately from the pool of available processes. When the client finishes processing, this server process either dies, or if reusable servers have been specified, it is returned to the pool.

You set up a pool of server processes by specifying the following parameters in the WS Server tab.

- Initial number of servers: The number of server processes that are prestarted for this workspace when the daemon starts up. These are available for use by new client processes with minimal initialization time. Instead of starting a new server process each time one is requested by a client, the daemon immediately allocates (to the client) a server from a pool of available servers. When the number of available server processes drops lower than the value specified in the Minimum number of available servers field, the daemon again starts server processes until the specified number of available servers is reached. The default for this parameter is 0, meaning that no servers are prestarted for this workspace.
- Minimum number of available servers: The minimum number of server processes in the prestarted server's pool before the Oracle Connect daemon resumes creating new server processes (up to the number specified in the Initial number of servers field value, described earlier). If this parameter is set to a value greater than that of the Initial number of servers field value, the daemon considers the value to be the same as the value specified in the Initial number of servers field. In this case, a new server process is started and added to the pool each time a server process is removed from the pool and allocated to a client). The default for this parameter is 0, which means that new servers are created only when there are no other available servers.
- Set maximum number of servers: The maximum number of available server processes pooled for this workspace. If the server is reusable, once a client disconnects from the server, the daemon returns the server to the pool of available servers. If the limit is reached, excess server processes are discarded.

Configuring a Binding Environment

Each binding configuration includes the following information:

- Environment settings, which are used to configure the environment used by any of the adapters defined in the binding.
- Application adapters on the current computer.

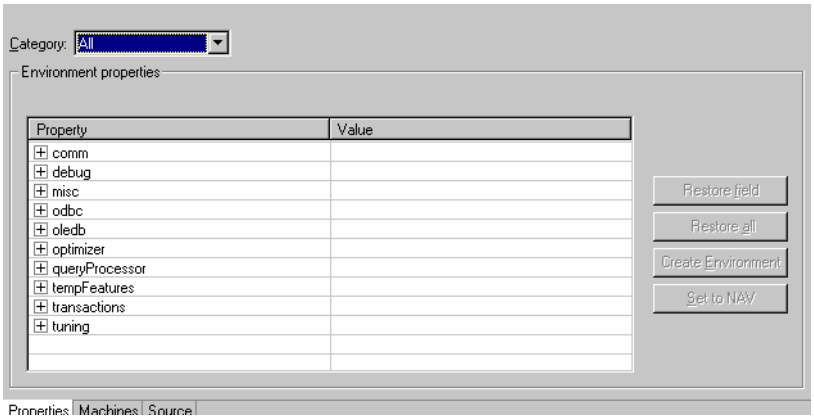
Configuring data sources and adapters is described in "Installing and Configuring OracleAS Adapters for Tuxedo".

To configure environmental settings in Oracle Studio, perform the following steps:

1. From the **Start** menu, select **Programs, Oracle**, and then select **Studio**.
2. In the Configuration Explorer, click the node next to the computer defined in "Configuring OracleAS Adapters for Tuxedo in Oracle Studio" on page 2-8.
3. Expand the **Bindings** node. The available binding configurations are listed.
4. Right-click **NAV** and select Edit Binding.
5. In the **Properties** tab edit the required environment settings as needed. To edit an environment setting, click the required property category node and then click the required value to edit.

   The binding **Properties** tab is displayed in the following figure:

Figure 6–2 The Binding Properties tab



The binding environment is divided into the following categories:

- comm Category
- debug Category
- miscellaneous Category
- odbc Category
- oledb Category
- optimizer Category
- queryProcessor Category
- transactions Category
- tuning Category

comm Category

The following table lists the parameters that define the communication buffers:

Table 6–2 comm Category Parameters

Parameter	Description
comCacheBufferSize	Specifies the size of a memory buffer on a client, which is used by the Oracle Connect client/server to store read-ahead data. The default is 200000 bytes.
comMaxSocketSize	Specifies the maximum bytes that can be written in one chunk on a socket. The default is -1 (no limitation).
comMaxXmlInMemory	Specifies the maximum size of an XML document held in memory. The default is 65535 bytes.
comMaxXmlSize	Specifies the maximum size of an XML document passed to another computer. The default is 65535 bytes.

debug Category

The following parameters define debugging and logging operations:

Table 6–3 debug Category Parameters

Parameter	Description
acxTrace	When set to true, the input xml sent to the back-end adapter and the output xml returned by the back-end adapter, are written to the log.
analyzerQueryPlan	This parameter is not applicable for use with OracleAS Adapter for Tuxedo.
gdbTrace	This parameter is not applicable for use with OracleAS Adapter for Tuxedo.
generalTrace	When set to true, logs general trace information. The default writes only error messages to the log.
logFile	The high-level qualifier of the log file for messages. The following types of message are written to the log: ■ Error messages. ■ Trace information and information about the query optimization strategy if generalTrace is set to true.
oledbTrace	This parameter is not applicable for use with OracleAS Adapter for Tuxedo.
optimizerTrace	This parameter is not applicable for use with OracleAS Adapter for Tuxedo.
queryWarnings	This parameter is not applicable for use with OracleAS Adapter for Tuxedo.
traceDir	This parameter is not applicable for use with OracleAS Adapter for Tuxedo.

miscellaneous Category

The following parameters define miscellaneous operations, including globalization support and the directory where temporary files are written.

Table 6–4 misc Category Parameters

Parameter	Description
`codepage`	For use with globalization support to identify the codepage for the workspace. See also: "Globalization Settings".
`cvtSeverityLevel`	The data type conversion policy when a conversion error occurs: ■ 0 (Default): The data in the output column will be a null or empty value. ■ 1: The data in the output column will be a null or empty value and the error is reported to the log. ■ 2: An error is reported and processing stops.
`edit`	This parameter is not applicable for use with OracleAS Adapter for Tuxedo.
`language`	Identifies the application language. A default codepage is selected based om the value specified for this parameter. See also: "Globalization Settings".
`nlsString`	Specifies the codepage used by a field whose data type is defined as `nlsString`. Use this for a field whose codepage is other than that of the computer codepage. This parameter includes the following values: ■ The name of the codepage. ■ Whether the character set reads from right to left (as in middle eastern character sets). The default is set to `false`.
`tempDir`	The directory where temporary files are written, including the temporary files created for use by hash joins and for sorting files. The default is the current high-level qualifier.
`year2000Policy`	Determines the way 2-digit years are converted into 4-digit years. When the parameter `year2000Policy` is not set, or when it is set to a value outside the range of values defined for the policy, as described in the following paragraphs, a default value of 5 and the Sliding Base Year policy is used. Two policies are provided: ■ Fixed Base Year: `year2000Policy` is set to a value greater than, or equal to 1900. In this case, the value of `year2000Policy` is the first 4-digit year after 1900 that can be represented by a 2-digit year. For example, if `year2000Policy` is set to 1905, the years 2000->2004 will be represented by 00->04. All other 2 digits will map to 19xx. This solution is most appropriate if there is live data at the low end (close to the year 1900), which the user wants to keep with the current 2-digit format. The user will probably change the base date only after ensuring that these old dates have been deleted from the data source. ■ Sliding Base Year: `year2000Policy` is set to a positive value less than 100. In this case, the value of `year2000Policy` represents the number of years ahead of the current year that can be represented by a 2-digit number. With each passing year the earliest year that can be represented by a 2-digit number changes to a year later.

odbc Category

The `odbc` parameters are not applicable for use with OracleAS Adapter for Tuxedo.

oledb Category

The `oledb` parameters are not applicable for use with OracleAS Adapter for Tuxedo.

optimizer Category

The `optimizer` parameters are not applicable for use with OracleAS Adapter for Tuxedo.

queryProcessor Category

The `queryProcessor` parameters are not applicable for use with OracleAS Adapter for Tuxedo.

transactions Category

The `transactions` parameters are not applicable for use with OracleAS Adapter for Tuxedo.

tuning Category

The `tuning` parameters are not applicable for use with OracleAS Adapter for Tuxedo.

Migration Considerations

You can migrate an adapter configuration from one platform to another. The configuration information is stored in the Oracle Connect repository on the source platform and is exported to an XML file which can then be imported to the target platform. Note that when migrating a configuration, any file names and paths that are specific to the source platform must be changed to valid files on the target platform.

Perform the following steps to migrate an adapter configuration using Oracle Studio:

1. From the **Start** menu, select **Programs, Oracle, and then select Studio**.
2. In the Configuration Explorer, right-click the required computer and select Export XML definitions.
3. Specify the path and name of the XML file, which stores the XML representation and complete configuration.
4. Edit any paths in the XML definition to the paths required on the target platform. For example, the setting for the `serverLogFile` property might need changing, depending on the platform.
5. Set up the target platform in Oracle Studio in the same way you set up the source platform, as described in "Configuring OracleAS Adapters for Tuxedo in Oracle Studio" on page 2-8.
6. In the Configuration Explorer, right-click the target computer and select Import XML definitions.
7. Import the XML file to the target platform.

Security Considerations

Oracle Connect works within the confines of the legacy platform security system. In addition, Oracle Connect provides the following security components:

- A binary XML encryption mechanism, which is activated as follows:
  1. The client's first message to the server includes a pre-defined shared key, includeing the key name and value in the connection string. The server gets the key value for the key name passed from the client from the native object store (NOS).
  2. The server generates a random 128-bit RC4 session key which is returned encrypted to the client, using the shared key. If no predefined shared key is provided, then a predefined, hardcoded key is used (this key is hardcoded on the client and on the server).
  3. Passwords are always encrypted when passed over the wire, using an RC4, 128-bit session key, regardless of whether the entire session is encrypted or not.
  4. If a predefined shared key was provided, then the entire session is encrypted. Otherwise, only the password exchange is encrypted (using the hardcoded key).
- Credentials: Passwords and usernames exchanged over the network are encrypted using a pre-defined, hardcoded, 128-bit RC4 session key.
- Design Time: Security within Oracle Studio to grant access to Oracle Studio itself and to grant access to computers, user profiles and workspaces.
- Runtime: Security used to access Tuxedo, including controlling the daemon for the access.

Setting Design Time Security

Setting design time security is described in the following sections:

- Securing access to Oracle Studio is described in "Setting Password Access to Oracle Studio" on page 2-9.
- Securing rights to configure a computer in Oracle Studio is described in "Specifying Users with Administrative Rights" on page 2-10.
- Securing access to user profiles is accomplished by right-clicking the relevant user profile in Oracle Studio and selecting Change Master Password. In the dialog box that is displayed, specify a password that must be provided in the future to edit the specific user profile.
- Securing access to workspaces is accomplished by right-clicking the relevant workspace in Oracle Studio and selecting Set Authorization. In the dialog box that is displayed, specify a valid user and password that must be provided in the future to edit the specific workspace.

Setting Run-Time Security

During runtime, security considerations are implemented as follows:

- When the client request accesses the legacy platform through the daemon, either anonymous access is allowed or a valid user name and password must be provided for the computer in the user profile. The userName and password

properties in the J2CA 1.5 Tuxedo adapter are used at this stage to access the daemon.

> **Note:** The user name used to access the daemon must also be the name of a user profile used.

- Access by the client must be through a valid port, according to the list of ports specified in the Workspace Access section of the **WS Security** tab in Oracle Studio. For details of the **WS Security** tab, refer to "WS Security" on page C-16.

> **Note:** Access to the legacy platform through a firewall using the NAT protocol is specified when the computer is added to Oracle Studio.

- To be allocated a server process, the client must be granted anonymous access to the workspace or be listed in the Workspace Users section of the **WS Security** tab in Oracle Studio. For details of the **WS Security** tab, refer to "WS Security" on page C-16.
- The ability to run commands on the daemon, such as starting or stopping a daemon or ending server processes is available only to administrators who have been registered in Oracle Connect as a daemon administrator. A client is registered as a valid daemon administrator in the **Daemon Security** tab in Oracle Studio, as described in "Daemon Security" on page C-5.

> **Note:** You can also specify administrators who can run commands only at the level of the workspace. Specify these administrators in the **WS Security** tab, as described in "WS Security" on page C-16.

Encrypting Network Communications

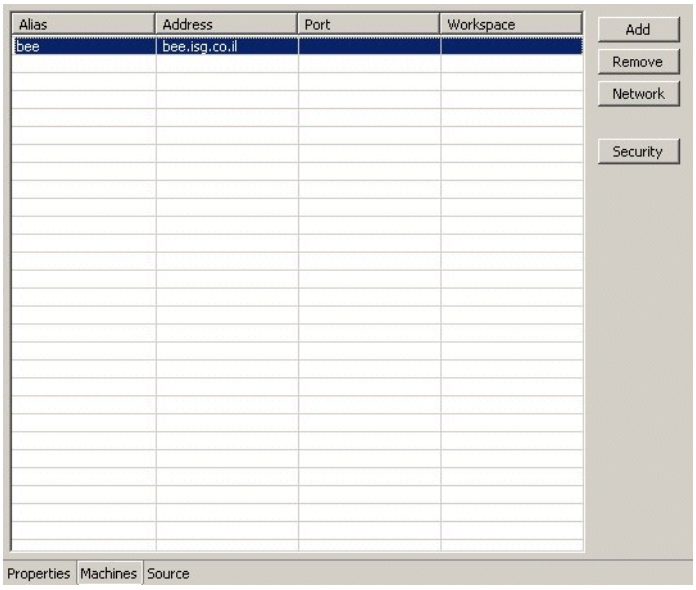
The encryption protocol between the client and the server is defined on the client machine.

Perform the following steps to encrypt network communications:

1. Right-click the binding in Oracle Studio Dessign Perspective Configuration explorer, and select Edit Binding.
2. Select the Machines tab.

   The Machines tab of the Binding editor is displayed, as shown in the following figure:

Figure 6–3 The Binding Editor Machines tab



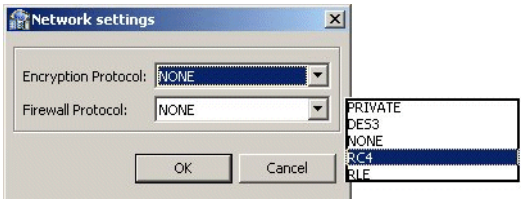
The image shows the Machines tab in the Binding editor.

****

3. Select the client machine, and click Network.

   The Network Settings screen is displayed, as shown in the following figure:

Figure 6–4 The Network Settings screen



The image shows the Network Settings screen, where you specify the encryption protocol for the network communications.

****

4. Select the required encryption protocol from the Encryption Protocol list, and then click OK.

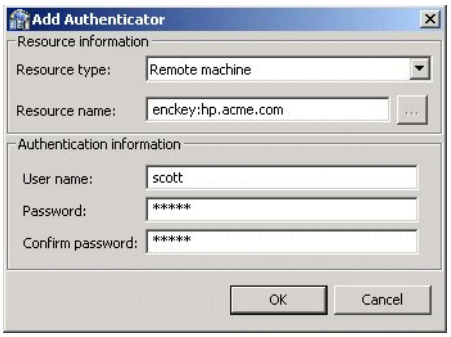
After specifying the encryption protocol, you must specify which servers the client is going to communicate with using the specified encryption protocol.

Perform the following steps to set the server computer for encrypted communications:

1. Right-click the user profile in the Configuration explorer and select **Edit User**.
2. In the User editor, select the client machine and click **Add**.

   The Add Authenticator screen is displayed, as shown in Figure 6–5.

Figure 6–5 The Add Authenticator screen



3. Configure the authenticator parameters as follows:
   - `Resource type`: Specify the resource type as Remote machine.
   - `Resource name`: Specify the communication to the machine is encrypted in the following format:

     `enckey: machine_name`

     where *`machine_name`* is the machine to which you are connecting.
   - `User name`: Specify the name associated with the encryption password and which the daemon on the remote machine looks up.

     Multiple clients may specify this name in their user profile. In this case, the user profile on the remote machine needs to list only this one username/password entry for network encryption (rather than listing and looking up multiple username/password pairs).

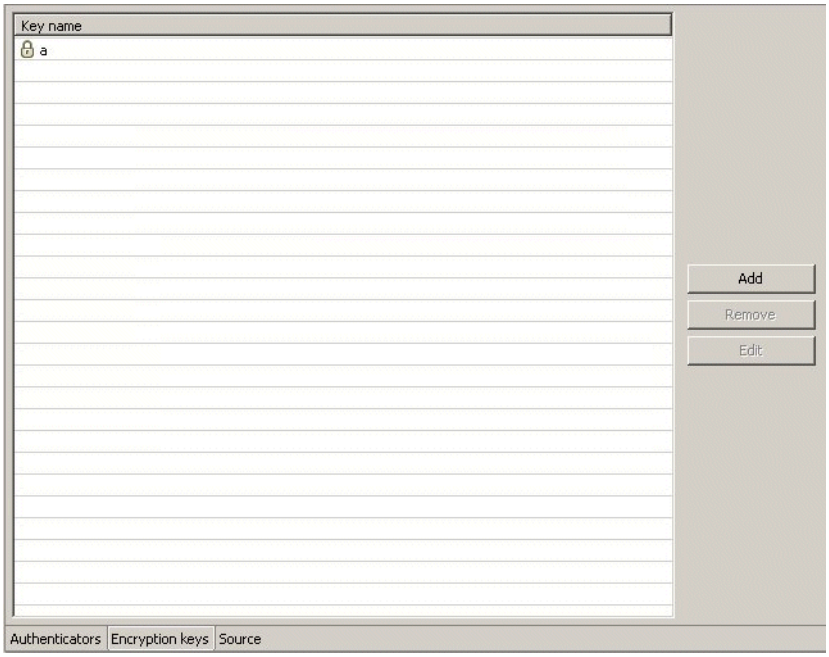
     If this user name entry is not specified, the daemon on the remote machine uses the name of the currently active user profile.
   - `Password`: Specify the password that is required in order to pass or access encrypted information over the network.
4. Click **OK**.

The server comupter must also be configured to decipher the encrypted information using an encryption key.

Perform the following steps to set the encryption key on the server comuter:

1. Select the server machine in the Attunity Studio Design perspective Configuration explorer.
2. Right-click the user profile and select **Edit User**. The User editor is displayed.
3. In the User editor, select the **Encryption Key** tab, shown in the following figure:

Figure 6–6 The Encryption Key Tab



4. Click **Add**.

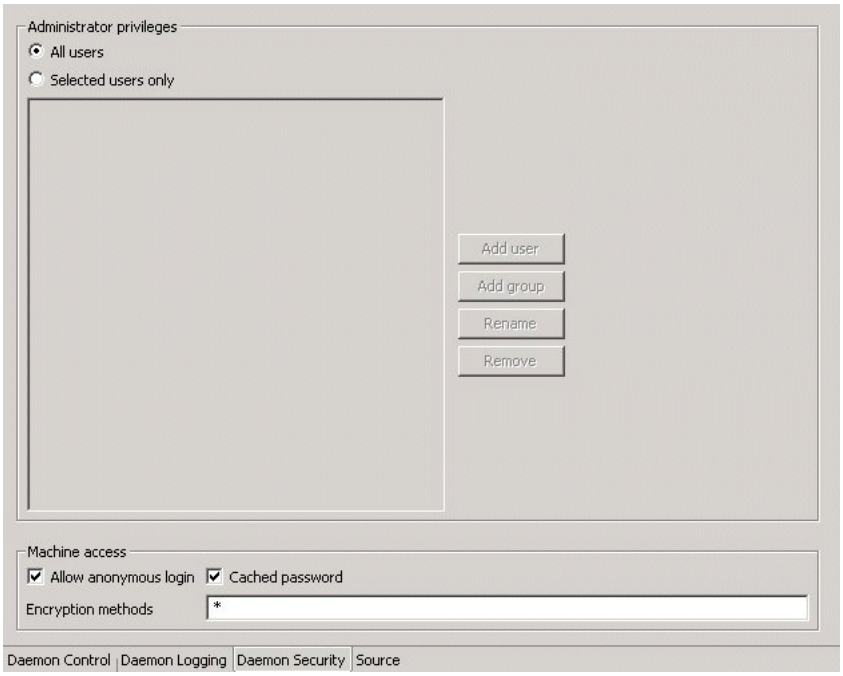
   The Add Encryption Key screen is displayed, as shown in the following figure:

Figure 6–7 The Add Encryption Key screen

5. Configure the encryption key parameters as follows:
   - `Key name`: Enter the name associated with the encryption password and which the daemon on this machine looks up.
   - `Key`: Enter the encryption key.
   - `Confirm key`: Re-enter the encryption key.
6. Click **OK**.
7. In the explorer tree, expand the Daemons node.
8. Right-click the daemon managing the connection and select **Edit Daemon**.
9. Select the **Daemon Security** tab.
10. In the Machine access area, enter **RC4** in the **Encryption methods** field.

    The Daemon Security tab is shown in the following figure:

Figure 6–8 Daemon Security Tab



Transaction Support

OracleAS Adapter for Tuxedo can participate in a distributed transaction, as the only one-phase commit resource. It supports local transactions by responding to ACX transactional verbs and issuing the corresponding ATMI transactional calls. The back-end service is responsible for the transaction semantics and implementation.

Transactional APIs included in the application, such as to start a transaction or commit a transaction, are mapped by OracleAS Adapter for Tuxedo to the corresponding Tuxedo transactional call. The adapter is also equipped with means to limit the Tuxedo transaction duration (timeout).

The following table maps the functionality of ACX transaction verbs to equivalent ATMI transaction calls.

Table 6–5 ACX-ATMI Transaction Correspondence.

ACX Transactional Verbs	ATMI Transactional Call	Comment
`transactionStart`	`tpbegin(lTimeout,0)`	Transaction duration can be specified within the adapter definitions.
`transactionCommit`	`tpcommit(0)`	
`transactionRollback`	`tpabort(0)`	

See Also: "Advanced Tuning of the Metadata" for further details about transaction support configuration parameters.

A

Advanced Tuning of the Metadata

Oracle Studio enables you to define adapter interactions such as Outbound and Inbound interactions. In addition, Oracle Studio defines input and output structures used for these interactions. The interactions and input and output structures are maintained as metadata in the **Metadata** tab of Oracle Studio.

Metadata for the Back-end Adapter

Using Oracle Studio, perform the following steps to maintain the metadata for Oracle AS Adapters for Tuxedo:

1. From the **Start** menu, select **Programs**, **Oracle**, and then select **Studio**.
2. Select the computer defined in "Configuring OracleAS Adapters for Tuxedo in Oracle Studio" on page 2-8.
3. Expand the **Bindings** node.
4. Expand the **NAV** binding node.
5. Expand the **Adapters** node to display the adapters.
6. Right-click the Tuxedo adapter in the Configuration Explorer and select **Edit Metadata** to open the **Metadata** tab.
7. Right-click the required interaction in the Metadata Explorer and select **Edit**.

The metadata editor opens, displaying the **General** tab, with general table details. The following tabs are used to view and edit the metadata:

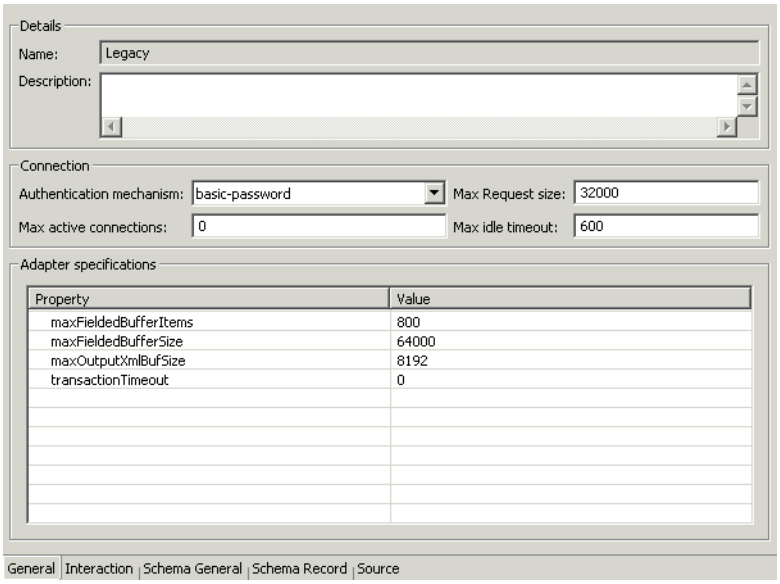
- General Tab
- Interaction Tab
- Schema General Tab
- Schema Record Tab
- Source Tab

General Tab

Use the **General** tab to maintain information describing the adapter and the connection to the adapter.

The **General** tab is shown in the following figure:

Figure A–1 The General tab



The **General** tab comprises fields, as listed in the following table:

Table A–1 General tab Components

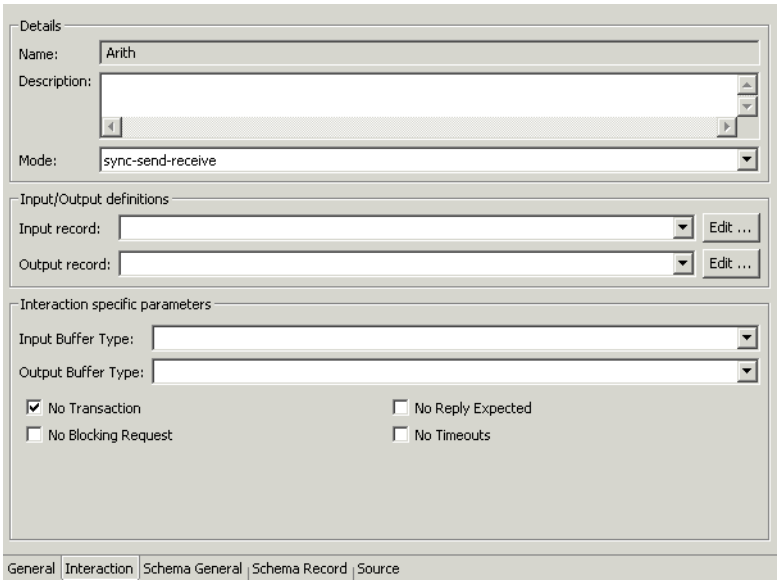
Field	Description
Adapter definition name	Specifies the name of the adapter definition.
Description	Specifies an identifying description of the adapter.
Authentication mechanism	Specifies the authentication to access the adapter. The available mechanisms are: kerbv5 none basic password
Max request size	Specifies the maximum size in bytes for an XML request or reply. Larger messages are rejected with an error.
Max active connections	Specifies the maximum number of simultaneous connections for an adapter (per process).
Max idle timeout	Specifies the maximum time, in seconds, that an active connection can stay idle. After that time, the connection is closed.
Adapter Specifications	Specifies the adapter-specific properties for an interaction.
maxFieldedBufferItems	Sets a limit to the number of items that a fielded buffer may hold. The default value is 800.
maxFieldedBufferSize	Limits the total size of a message buffer of type FML. The default is set to 2 Kilobytes.
maxOutputXmlBufSize	Controls the size of memory allocated for output message buffers of type XML. The default is set to 8 Kilobytes.
transactionTimeout	Controls the duration of a Tuxedo transaction (in seconds). If not set (or set to 0), the Tuxedo transaction will not be interrupted and will last until explicitly terminated. This is the default.

Interaction Tab

Use the **Interaction** tab to define the general details of the interaction in addition to its input and output definitions.

The **Interactions** tab is shown in the following figure:

Figure A–2 The Interactions tab



The **Interaction** tab comprises fields, as listed in the following table:

Table A–2 Interaction tab Components

Field	Description
Interaction name	Specifies the name of the interaction.
Description	Provides a descriptive identifier for the interaction.
Mode	Determines the interaction mode. The following interaction modes are available: sync-send-receive: The interaction sends a request and expects to receive a response. sync-send: The interaction sends a request and does not expect to receive a response. sync-receive: The interaction expects to receive a response. async-send: Not applicable.
Input record	Identifies an input record.
Output record	Identifies an output record for the results of an interaction.

Table A–2 (Cont.) Interaction tab Components

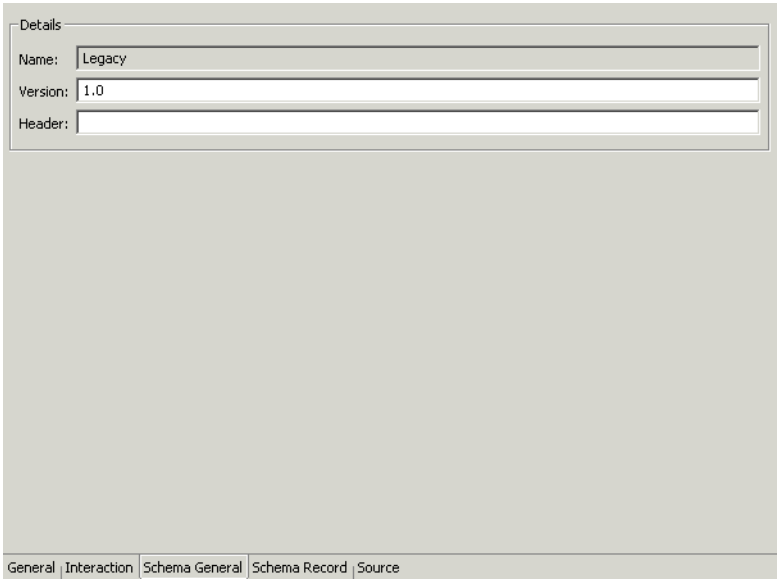
Field	Description
Interaction Specific Parameters	Defines the properties and values of parameters specific to an interaction. The following properties are available: Input Buffer Type: The type of buffer used for the input. Output Buffer Type: The type of buffer used for the results of an interaction. No Transaction: Enables a service to be executed, regardless of the transaction context. No Reply Expected: For future use. No Blocking Request: Avoids a `FROM` request submission if a blocking condition exists. No Timeouts: Ignores blocking timeouts.

Schema General Tab

Use the **Schema General** tab to define the general details of the input and output record structures for the interaction.

The **Schema General** tab is shown in the following figure:

Figure A–3 The Schema General tab



The **Schema General** tab comprises fields, as listed in the following table:

Table A–3 Schema General tab Components

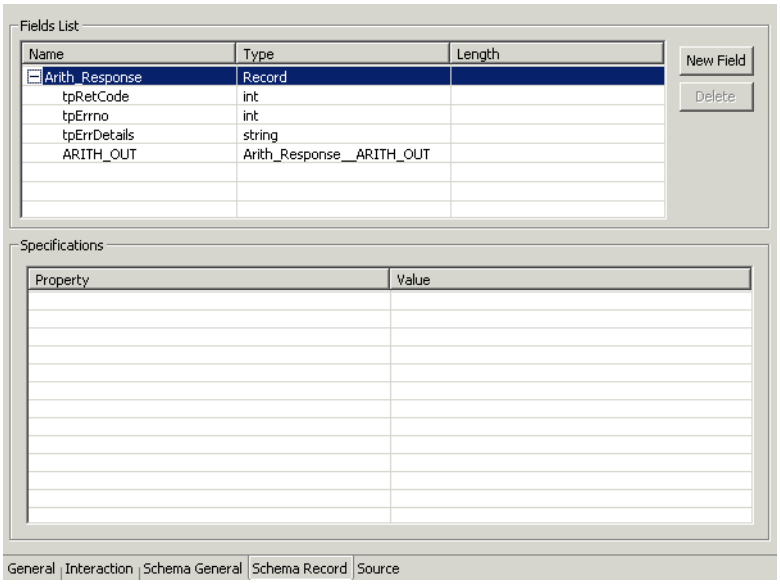
Field	Description
Schema name	The name of the adapter.
Version	The schema version.
Header	A `C` header file to map between the data structure and the adapter.

Schema Record Tab

Use the **Schema Record** tab to define the input and output record structures for the interaction.

The **Schema Record** tab is shown in the following figure:

Figure A–4 The Schema Record tab



The **Schema Record** tab comprises fields, as listed in the following table:

Use the Fields List area to define single data items within a record.

Table A–4 Schema Record tab Components

Field	Description
Name	Specifies the name of the field.
Type	The data type of the field. The following are valid data types: ■ `Binary` ■ `Boolean` ■ `Date` ■ `Double` ■ `Float` ■ `Int` ■ `Long` ■ `Numeric` ■ `Short` ■ `String` ■ `Time` ■ `Timestamp` ■ `XML`
Length	The size of the field including a null terminator, when the data type supports null termination.

See Also: "OracleAS Adapters for Tuxedo Message Buffer Support and Data Type Support" for details about the mapping from COBOL data types to Oracle Connect data types.

Note: Use the Specifications box to specify field properties.

Source Tab

The **Source** tab displays the XML representation of the adapter metadata.

A sample **Source** tab is displayed in the following figure:

Figure A–5 The Source tab

```
<?xml version="1.0" encoding="UTF-8"?>
<table name="customer" datasource="___NAVDEMO" description=""
    fileName="D:\Program Files\Oracle\Server\demo\customer"
    nBlocks="0" nRows="0" bookmarkSize="4" organization="index">
    <fields>
        <field name="c_custkey" datatype="int4"/>
        <field name="c_name" datatype="cstring" size="25"/>
        <field name="c_address" datatype="cstring" size="40"/>
        <field name="c_nationkey" datatype="int4"/>
        <field name="c_phone" datatype="string" size="15"/>
        <field name="c_acctbal" datatype="double"/>
        <field name="c_mktsegment" datatype="string" size="10"/>
        <field name="c_comment" datatype="cstring" size="117" nullable="true"/>
    </fields>
    <keys>
        <key name="cindex" size="4">
            <segments>
                <segment name="c_custkey"/>
            </segments>
        </key>
    </keys>
</table>
```

B

OracleAS Adapters for Tuxedo Message Buffer Support and Data Type Support

This appendix contains the following sections:

OracleAS Adapters for Tuxedo Message Buffer Support

Oracle Connect provides support for all standard types of OracleAS Adapters for Tuxedo message buffers as I/O, as follows:

- `STRING`: A null terminated character array. The data type is character and its length is determined by counting the chracters in the buffer until reaching the null character. It is commonly used by C programs.
- `CARRAY`: An array of un-interpreted arbitrary binary data. The application must specify the buffer length for `CARRAY` message buffers when used as input to ATMI functions.
- `XML`: An XML formatted data. This buffer type enables Tuxedo applications to use XML for exchanging data within and between applications. Tuxedo applications can send and receive simple XML buffers, and route them to the appropriate servers. Data dependent routing is supported for this buffer type.
- `VIEW` (16-bit): A C structure layout. This buffer is used for fixed collections of data elements, structures or records. VIEW records support integral data types such as long integer, character, and decimal. `VIEW` records do not support structures within structures, nor do they support arrays of structures or pointers.
- `VIEW` (32-bit): A C structure layout where 32-bit `FML` identifiers are used.
- `FML` (16-bit): An abstact data type, used to create, access, modify and delete fields. It is a data structure that stores tagged values. Values are typed, can be specified more than once, and vary in length. Additionally, `FML` buffers support storage of more than one value for a field. The variable length format of feilded buffers enables multiple field occurences to be stored and retrieved.
- `FML` (32-bit): An FML type where 32-bit `FML` identifiers are used.

> **Note:** Synonyms such as X_C_TYPE,and X_OCTET are also recognized.

Unstructured message buffers are wrapped within a record as follows:

- A message buffer of type STRING is wrapped within a record containing a single field of type `string` with a fixed size.
- A message buffer of type CARRAY is wrapped within a record containing a single field of type `binary` with a fixed size.
- A message buffer of type XML is wrapped within a record containing a single field of type XML.

User-Defined Message Buffers

Oracle Connect does not support user-defined message buffers.

Data Type Support

This section contains the following topics:

Data Type Mapping

OracleAS Adapters for Tuxedo support a number of data types that are used to define metadata in Oracle Studio. The data types are mapped from the Tuxedo data types during the import procedure.

> **Note:** The mapping of data types between OracleAS Adapters for Tuxedo and Oracle Application Server is performed internally by Oracle Connect.

Table B–1 Data Type Mapping: Tuxedo and Oracle Connect

Tuxedo Data Type	Oracle Connect Data Type
carray	binary
char	string
double	double
float	double
int	int
long	int
short	int
string	string

Data Type Handling

OracleAS Adapters for Tuxedo handle the fields contained in the VIEW file as follows:

- Specifying a value greater than one in the `COUNT` field of the Tuxedo field definition within a VIEW file, translates to array=xx attribute.
- Specifying a value in the `SIZE` field of Tuxedo field definition within a VIEW file, translates to size=yy attribute.

- Specifying C in the FLAG field of Tuxedo field definition within a VIEW file, imposes generation of an extra leading COUNTER field to hold the actual count value.
- Specifying L in the FLAG field of Tuxedo field definition within a VIEW file, imposes generation of an extra leading LENGTH field to hold the actual length value.

Note:

- The L flag is applicable for STRING and CARRAY data types only.
- In the case where a COUNT value greater than one is specified for the field, the extra leading LENGTH field is generated as an array. The array entries should hold the actual length values of the corresponding field array entries.

- Specifying a value in the NULL field of Tuxedo's field definition within a VIEW file is not reflected in Oracle Connect back-end adapter record.

Tuxedo's restrictions with regard to VIEW fields are preserved.

The following is an example of a VIEW file definition:

```
VIEW emp
#TYPE     CNAME            FBNAME          COUNT   FLAG  SIZE   NULL
long      lSalary          SALARY            1      -      -     0
short     nDeptnum         DEPTNUM           1      -      -     0
short     nEmpnum          EMPLOYEE_NUMBER   1      -      -     0
short     nJobcode         JOBCODE           1      -      -     0
string    szMessageText    MESSAGE_TEXT      1      -     80     ""
string    szFirstName      FIRST_NAME        1      -     16     ""
string    szLastName       LAST_NAME         1      -     21     ""
END
```

The following is an example of a field table that defines the FML field IDs:

```
*base 100
# name           number   type    flags comments
MESSAGE_TEXT          1   string   -     -
#
DEPTNUM             100   short    -     -
EMPLOYEE_NUMBER     101   short    -     -
JOBCODE             102   short    -     -
FIRST_NAME          103   string   -     -
LAST_NAME           104   string   -     -
SALARY              105   long     -     -
```

Header Record Structure

Each adapter has a HEADER record which is mapped to Tuxedo ATMI Queue Control Structure. The HEADER record is included in all records defined.

HEADER input fields (for writing and subscribing) are as listed in the following table:

Table B–2 HEADER Record Input Fields

Field	Description
long flags;	Indicates which of the values are set.
long deq_time;	Indicates absolute/relative time for dequeing.
long priority;	Specifies the enqueing priority.
long exp_time;	Specifies the expiration time.
long delivery_qos;	Specifies the delivery quality of service.
long reply_qos;	Specifies the reply quality of service.
long urcodes;	User-return code.
char corrid[32];	Specifies the correlation identifier used to identify the message.
char replyqueue[16];	Species the queue name for the reply message.
char failurequeue[16];	Specifies the queue name for failure messages.

HEADER output fields (for reading) are as listed in the following table:

Table B–3 HEADER Record Output Fields

Field	Description
long flags;	Indicates which of the values should be set.
long priority;	Specifies the enqueing priority.
char msgid[32;]	Specifies the ID of the message dequeued.
char corrid[32];	Specifies the correlation identifier used to identify the message.
long delivery_qos;	Specifies the delivery quality of service.
long reply_qos;	Specifies the reply quality of service.
char replyqueue;	Specifies the queue name for the reply.
char failurequeue[16];	Specifies the queue name for failure messages.
long diagnostic;	Specifies the reason for failure.
long appkey;	Specifies the application authentication client key.
long urcodes;	User-return code.
CLIENTID cltid;	Specifies the client identifier for the originating client.
char replyqueue[16];	Species the queue name for the reply message.
char failurequeue[16];	Specifies the queue name for failure messages.

C

Advanced Tuning of the Daemon

The daemon configuration is managed using Oracle Studio. Daemon configuration is divided into the following groups:

- Daemon Control
- Daemon Logging
- Daemon Security
- Workspaces

Daemon Control

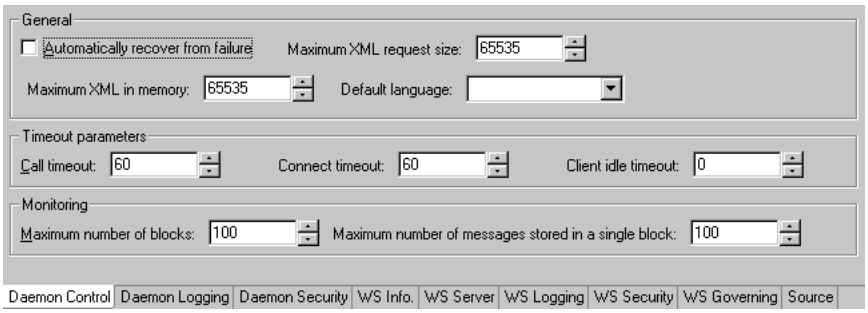
Using the **Daemon Control** tab, you define various daemon control options. The **Daemon Control** tab is accessed as follows:

1. From the **Start** menu, select **Programs, Oracle**, and then select **Studio**.
2. Select the required computer from the Configuration Explorer.
3. Right-click the computer and select **Open Runtime Perspective**.
4. Right-click the required daemon in the Runtime Explorer and select **Edit Daemon Configuration**. The **Daemon Control** tab is displayed.
5. After making changes to the daemon, right-click the daemon and select **Reload Configuration**.

> **Note:** You can also change daemon settings using the Configuration Explorer, by selecting a computer and scrolling the list to the required daemon. Right-click the daemon and select **Edit Daemon**.
>
> Changes made to the daemon configuration are only implemented after the configuration is reloaded using the **Reload Configuration** option in the Runtime Manager perspective.

The **Daemon Control** tab is shown in the following figure:

Figure 6–9 The Daemon Control tab



The **Daemon Control** tab comprises fields, as listed in the following table:

Table 6–6 Daemon Control tab Components

Field	Description
Automatically recover from failure	The daemon restarts automatically if it fails for any reason (any error that causes the daemon process to terminate, such as network process lost or the CPU running the daemon crashes and the backup daemon is defined on another CPU). All available and unconnected servers are terminated and any connected servers are marked and terminated on release. Also the backup starts a backup for itself. The backup appends a new log file to the log of the original daemon, adding a line indicating that a backup daemon was started.
Maximum XML request size	The maximum number of bytes that the daemon handles for an XML document.
Maximum XML in memory	The maximum amount of space reserved for the XML in memory.
Default language	The language that the daemon supports. This setting is used when working with a client with a code page different from the server code page.
Call timeout	The timeout period for *short* calls for all daemons. The definition of a short call is a call that should be completed in a few seconds. For example, most calls to a database such as `DESCRIBE` should be completed in a few seconds as opposed to call like a `GETROWS` call, which can take a long time. In heavily loaded or otherwise slow systems, even short calls such as calls to open a file, may take a significant amount of time. If a short call takes more than the specified time to complete, then the connection is stopped. The default value for this parameter is 60 seconds. Values of less than 60 seconds are considered to be 60 seconds. Specifying the timeout in a workspace overrides the value set in this field for that workspace.

Table 6–6 (Cont.) Daemon Control tab Components

Field	Description
Connect timeout	The time the client waits for a daemon server to start. If the daemon server does not start within this period, then the client is notified that the server did not respond. The value specified for this parameter serves as the default timeout for all the workspaces listed in the daemon configuration. The default value for this parameter is 60 seconds. **Notes**: ■ Entering the timeout in a workspace overrides the value set in this field for that workspace. ■ Even if the XML source does not list this parameter in the workspace section, the workspace gets it using the default value. If you want to prevent a workspace from using the default value, you must enter a value of zero for this parameter in the workspace section.
Client idle timeout	The maximum amount of time any daemon client may be idle before the connection with the server is closed. Specifying the timeout in a Workspace overrides this setting for that workspace.

Daemon Logging

Using the **Daemon Logging** tab, you define the daemon log file settings, the log file structure and the location where the log is saved. In addition, use it to define the data that is logged and traced in the file.

The **Daemon Logging** tab is accessed as follows:

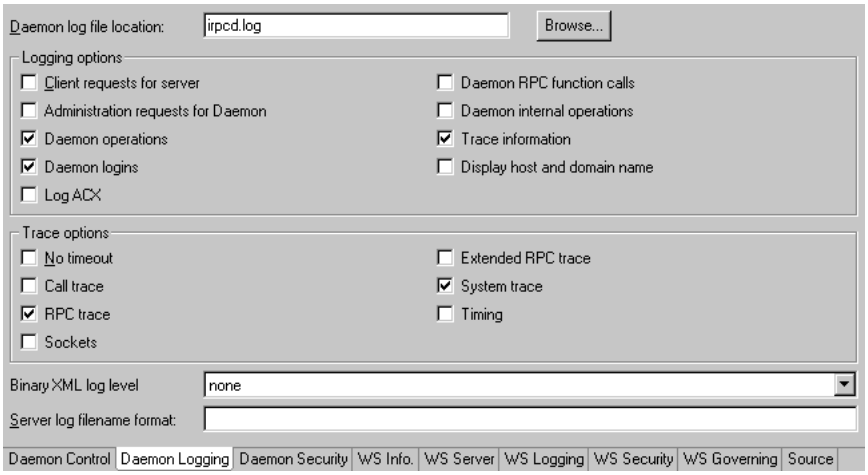
1. From the **Start** menu, select **Programs**, **Oracle**, and then select **Studio**.
2. Select the required computer from the Configuration Explorer.
3. Right-click the computer and select **Open Runtime Perspective**.
4. Right-click the daemon in the Runtime Explorer and select **Edit Daemon Configuration**.
5. Click the **Daemon Logging** tab.
6. After making changes to the daemon, right-click the daemon and select **Reload Configuration**.

> **Note:** You can also change daemon settings using the Configuration Explorer, by selecting a computer and scrolling the list to the required daemon. Right-click the daemon and select **Edit Daemon**.
>
> Changes made to the daemon configuration are only implemented after the configuration is reloaded using the **Reload Configuration** option in the Runtime Manager perspective.

7. Right-click the daemon and select **End Unused Servers**. Any servers in the connection pool are closed and new servers start with the new configuration.

The **Daemon Logging** tab is shown in the following figure:

Figure 6–10 The Daemon Logging tab



The **Daemon Logging** tab comprises fields, as listed in the following table:

Table 6–7 Daemon Logging tab Components

Field	Description
Daemon log file location	Specifies the daemon produces its log data. The full path must be specified.
Logging options	Specifies what tracing is performed.
Client requests for server	Logs client requests for server activations; this provides logging of the process IDs of the started servers along with the location of the log files.
Administration requests for daemon	Logs all of the administration requests for the daemon.
Daemon operations	Logs all of the daemon operations.
Daemon logins	Logs daemon logins.
Daemon RPC function calls	Logs all daemon RPC function calls.
Daemon internal operations	Logs daemon internal operations.
Log trace information	Logs low-level RPC operations.
Display host and client domain name	Specifies whether the client host and domain name are logged rather than the client IP address. The default is false.
Trace options	Specifies the type of tracing being performed.
No timeout	Disables the standard RPC timeouts, setting them to a long duration (approximately an hour) to facilitate debugging.
Call trace	Generates a message in the server log file for each RPC function called. This is useful for troubleshooting the server.
RPC trace	Enables debugging messages on the server.
Sockets	Generates a message in the server log file for each socket operation.
Extended RPC trace	Generates a verbose message in the server log file for each low-level RPC function called. This is useful for troubleshooting the server.

Table 6–7 (Cont.) Daemon Logging tab Components

Field	Description
System trace	Generates system-specific tracing of various operations.
Timing	Generates a timestamp for every entry to the server log file.
Binary XML log level	Sets the binary XML log level. Your options are: ■ debug ■ none (the default) ■ api ■ info
Server log filename format	Defines the name and location of the server log file. The field must specify the full path name. If no directory information is provided for the log file, then it will be located in the login directory of the account running Oracle Connect workstation.

The following tokens can appear in the log file template and will be replaced accordingly:

- `%A`: workspace name
- `%D`: date (yymmdd)
- `%I`: instance number of the given workspace server
- `%L`: server account login directory
- `%P`: server process ID
- `%T`: time (hhmmss)
- `%U`: server account name (username)

For example, `%L/server_%A%I.log` may produce a log file such as: `/usr/smith/server_sales15.log`.

The default log file template is `%L/server_%A%I.log`.

Daemon Security

The **Daemon Security** tab is used to:

- Grant administration rights for the daemon.
- Determine access to the computer.

The **Daemon Security** tab is accessed as follows:

1. From the **Start** menu, select **Programs, Oracle**, and then select **Studio**.
2. Select the required computer from the Configuration Explorer.
3. Right-click the computer and select **Open Runtime Perspective**.
4. Right-click the daemon in the Runtime Explorer and select **Edit Daemon Configuration**.
5. Click the **Daemon Security** tab.
6. After making changes to the daemon, right-click the daemon and select **Reload Configuration**.

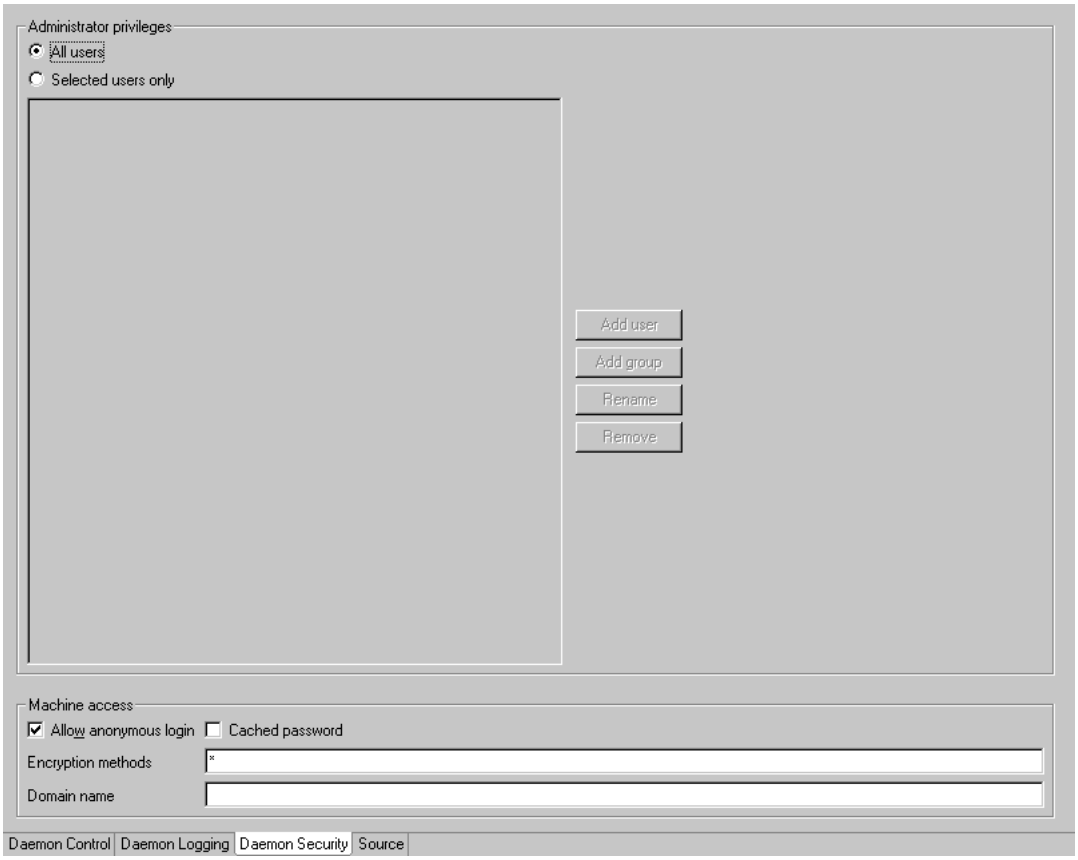
Note: You can also change daemon settings using the Configuration Explorer, by selecting a computer and scrolling the list to the required daemon. Right-click the daemon and select **Edit Daemon**.

Changes made to the daemon configuration are not implemented. They are only implemented after the configuration is reloaded using the **Reload Configuration** option in the Runtime Manager.

7. Right-click the daemon and select **End Unused Servers**. Any servers in the connection pool are closed and new servers start with the new configuration.

The **Daemon Security** tab is shown in the following figure:

Figure 6–11 The Daemon Security tab



The **Daemon Security** tab comprises fields, as listed in the following table:

Table 6–8 Daemon Security tab Components

Field	Description
Administrators privileges	Identifies the users (accounts) allowed to perform administrative tasks (tasks that require administrative login).
All users	Enables all users to access the daemon and change the settings.

Table 6–8 (Cont.) Daemon Security tab Components

Field	Description
Selected users only	Identifies the names of users (accounts) and groups that can be administrators.[1] If a user is not specified, the account from which the daemon was started is considered the administrator. Note that the daemon does not require the user to log in to the account on the system, but to log in to the daemon using the account name and password.
Machine access	Manages access to the computer.
Allow anonymous login	Whether workspaces allow anonymous logins (without user name/password entries). For the optimal level of security, keep this option unchecked and define a username for the Daemon Administrators parameter. If unchecked, then no workspace can have an anonymous client. If checked, then a particular workspace allows anonymous clients.
Cached password	Enables login passwords to be cached. This enhances performance by reducing login times for future connections from the same client in a session.
Encryption methods	Specifies the encryption method used to send information across the network. The default is an asterisk (*), meaning that all methods are acceptable. If an encryption method is specified, it must be used. The RC4 and DES3 protocols are currently supported.

[1] The name is prefixed with '@', to utilize the operating system GROUP feature.

Workspaces

A daemon can include a number of workspaces. A workspace defines the server processes and environment that are used for the communication between the client and the server for the duration of the client request. Each workspace has its own definition. The workspace definition is divided into the following groups:

- WS Info.
- WS Server
- WS Logging
- WS Security
- WS Governing: This tab is not applicable for use with OracleAS Adapters for Tuxedo

WS Info.

Using the **WS Info**. tab, you specify the features that control the operation of the workspace, such as the server type, the command procedure used to start the workspace and the binding configuration associated with this workspace.

The **WS Info**. tab is accessed as follows:

1. From the **Start** menu, select **Programs, Oracle**, and then select **Studio**.
2. Select the required computer from the Configuration Explorer.
3. Right-click the computer and select **Open Runtime Perspective**.

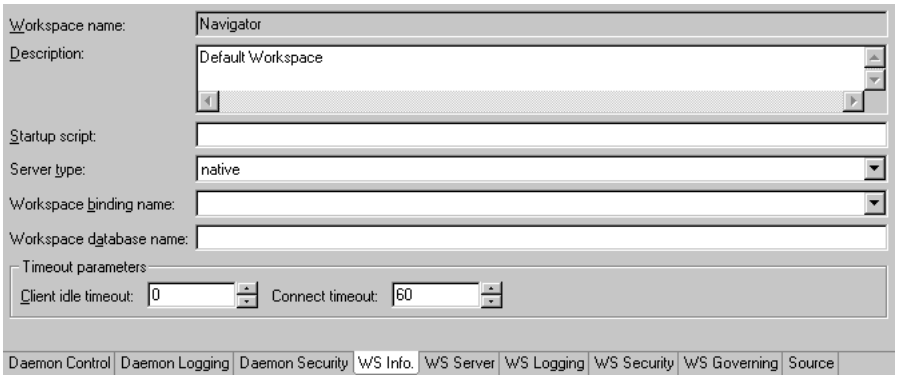
4. Expand the Daemons node to display the workspaces in the Runtime Explorer.
5. Right-click the workspace and select **Edit Workspace Configuration**. The WS Info. tab opens.
6. After making changes to the workspace, right-click the daemon and select **Reload Configuration**.

> **Note:** You can also change daemon settings using the Configuration Explorer, by selecting a computer and scrolling the list to the required daemon. Right-click the daemon and select **Edit Daemon**.
>
> Changes made to the daemon configuration are not implemented. They are only implemented after the configuration is reloaded using the **Reload Configuration** option in the Runtime Manager.

7. Right-click the daemon and select **End Unused Servers**. Any servers in the connection pool are closed and new servers start with the new configuration.

The **WS Info**. tab is shown in the following figure:

Figure 6–12 The WS Info tab



The **WS Info**. tab comprises fields, as listed in the following table:

Table 6–9 WS Info tab Components

Field	Description
Workspace name	The name used to identify the workspace. **Note**: The default configuration includes the default Navigator workspace. This workspace is automatically used if a workspace is not specified as part of the connection settings.
Description	A description of the workspace.
Startup script	The full path name of the script that starts the workspace server processes. The script specified here must always activate the `nav_login` procedure and then run the server program (svc). If you do not specify the directory, the startup procedure is taken from the directory where the daemon resides. Oracle Connect includes a default startup script, which it is recommended to use.

Table 6–9 (Cont.) WS Info tab Components

Field	Description
Server type	This field is not applicable for use with OracleAS Adapters for Tuxedo.
Workspace binding name	This field is not applicable for use with OracleAS Adapters for Tuxedo.
Timeout parameters	The time the client waits for the workspace server to start. If the workspace server does not start within this period, then the client is notified that the server did not respond. Specifying the timeout here overrides the default setting, specified in the Control section. See Also: "Daemon Control" on page C-1 for details about the Daemon Control section.
Client idle timeout	The maximum amount of time a workspace client can be idle before the connection with the server is closed.
Connect timeout	The time the client waits for a workspace server to start. If the workspace server does not start within this period, then the client is notified that the server did not respond.

WS Server

Using the **WS Server** tab, you specify the features that control the operation of the servers started up by the workspace and allocated to clients.

For example, you can configure the workspace to start up a number of servers for future use, prior to any client request, instead of starting each server when a request is received from a client.

The **WS Server** tab is accessed as follows:

1. From the **Start** menu, select **Programs, Oracle,** and then select **Studio**.
2. Select the required computer from the Configuration Explorer.
3. Right-click the computer and select **Open Runtime Perspective**.
4. Expand the Daemons node to display the workspaces in the Runtime Explorer.
5. Right-click the workspace and select **Edit Workspace Configuration**.
6. Click the **WS Server** tab.
7. After making changes to the workspace, right-click the daemon and select **Reload Configuration**.

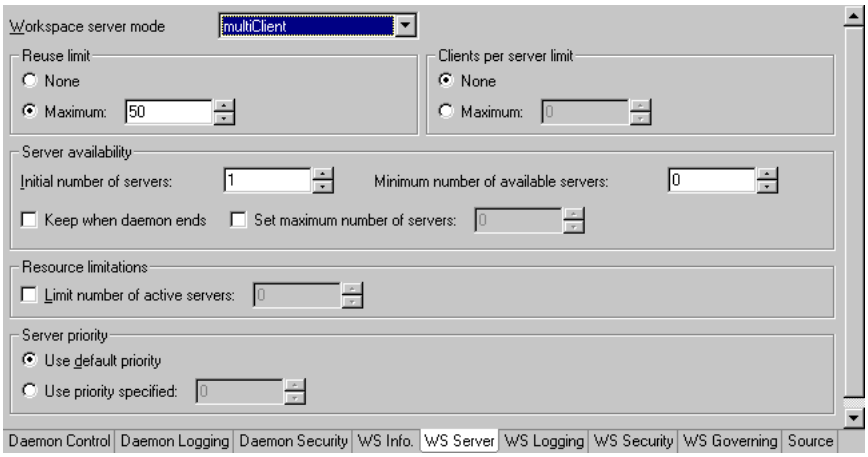
Notes:

- You can also change daemon settings using the Configuration Explorer, by selecting a computer and scrolling the list to the required daemon. Right-click the daemon and select **Edit Daemon**.
- Changes made to the daemon configuration are not implemented. They are only implemented after the configuration is reloaded using the **Reload Configuration** option in the Runtime Manager.

8. Right-click the daemon and select **End Unused Servers**. Any servers in the connection pool are closed and new servers start with the new configuration.

The **WS Server** tab is shown in the following figure:

Figure 6–13 The WS Server tab



The **WS Server** tab comprises fields, as listed in the following table:

Table 6–10 WS Server tab Components

Field	Description
Workspace server mode	Specifies the type of new server processes that the daemon starts up. The daemon supports the following server modes: ■ `singleClient`: Each client receives a dedicated server process. The account in which a server process runs is determined either by the client login information or by the specific server workspace. This mode enables servers to run under a particular user account and isolates clients from each other, as each receives its own process. However, this server mode incurs a high overhead due to process startup times and can use a lot of server resources as it requires as many server processes as concurrent clients. ■ `multiClient`: Clients share a server process and are processed serially. This mode has low overhead because the server processes are already initialized. However, because clients share the same process, they can impact one another, especially if they issue lengthy queries. The number of clients that share a process is determined by the Clients per server limit field. ■ `multiThreaded`: This mode is not applicable for use with OracleAS Adapter for Tuxedo. ■ `reusable`: An extension of single-client mode. Once the client processing finishes, the server process does not die and can be used by another client, reducing startup times and application startup overhead. This mode does not have the high overhead of single-client mode because the servers are already initialized. However, this server mode can use a lot of server resources as it requires as many server processes as concurrent clients. **Note**: The other modes can be set so that the server processes are reusable. The number of times a process can be reused is controlled by the Reuse limit field value.
Reuse limit	Sets the maximum number of times a particular server can be reused. A one-client server can be reused after its (single) client has disconnected. Reuse of servers enhances startup performance because it avoids the need to repeat initialization. The default for this field is none (0), indicating that server reuse is unlimited. This parameter is disabled only if the server mode value is `singleClient`.
Clients per server limit	Sets the maximum number of clients a server process for the current workspace accepts. The default for this field is none (0), indicating that the number of clients for each server is unlimited. This field is enabled only if the server mode value is `multiClient` or `multiThreaded`.

Table 6–10 (Cont.) WS Server tab Components

Field	Description
Server availability	Specifies the number of servers in a pool of servers, available to be assigned to a client. The following options are available: ■ Initial number of servers: The number of server processes that are prestarted for this workspace when the daemon starts up. When the number of available server processes drops lower than the value specified in the Minimum number field, the daemon again starts server processes until this number of available server processes is reached. The default for this field is 0. ■ Minimum number: The minimum number of server processes in the prestarted pool before the daemon resumes creating new server processes (to the value specified in the Initial number of servers field). If this field is set to a value higher than the Initial number of servers field, the daemon uses the value specified in the Initial number of servers field. The default for this field is 0. ■ Keep when daemon ends: When a daemon is shutdown, all the servers started by that daemon are also killed, even if they are active. Set this field to `true` if you want the servers for the workspace to remain active, even after the daemon has been shut down. If this field is set to `true`, it is the responsibility of the system operator or manager to ensure that the servers are eventually killed. This must be done at the system level. ■ Set maximum number of servers: The maximum number of available server processes. Once this number is reached, no new nonactive server processes are created for the particular workspace. For example, if a number of server processes are released at the same time, so that there are more available server processes than specified by this field, the additional server processes higher than this value are terminated. The default for this field is zero, meaning that there is no maximum.

Table 6–10 (Cont.) WS Server tab Components

Field	Description
resource limitations	Specifies the number of servers that can be in use at any one time. The more servers used, the greater the system resources that are used. The following options are available: ■ Number of subtasks: The number of subtasks for a server that are prestarted for this workspace when the daemon starts up. Thus, setting 10 prestarted servers and 10 subtasks results in 100 tasks started (10 subtasks for each process). ■ Limit number of active servers: The maximum number of active server processes (either available or in use). Once reached, no new server processes will be created for the particular workspace and client connections would be rejected if there is no available server to accept them. Once the number of active servers drops below the maximum (for example, a client disconnects from a server and the server terminates), new servers can again be started. If the value of this field is set to a nonzero value lower than the value for the Initial number of servers field, the daemon assumes it is set to the same value as the Initial number of servers field. The default for this field is 0, meaning that no maximum is enforced.
Server Priority	The priority for servers. For example, a workspace for applications with online transaction processing can be assigned a higher priority than a workspace that requires only query processing. The following priority options are available: ■ Use default priority: Sets the priority as 0. There is no specific priority for this workspace. ■ Use priority: Enables setting the priority.

WS Logging

Using the **WS Logging** tab, you specify parameters to log, that occur with the workspace server process.

The **WS Logging** tab is accessed as follows:

1. From the **Start** menu, select **Programs, Oracle**, and then select **Studio**.
2. Select the required computer from the Configuration Explorer.
3. Right-click the computer and select **Open Runtime Perspective**.
4. Expand the Daemons node to display the workspaces in the Runtime Explorer.
5. Right-click the workspace and select **Edit Workspace Configuration**.
6. Click the WS Logging tab.
7. After making changes to the workspace, right-click the daemon and select **Reload Configuration**.

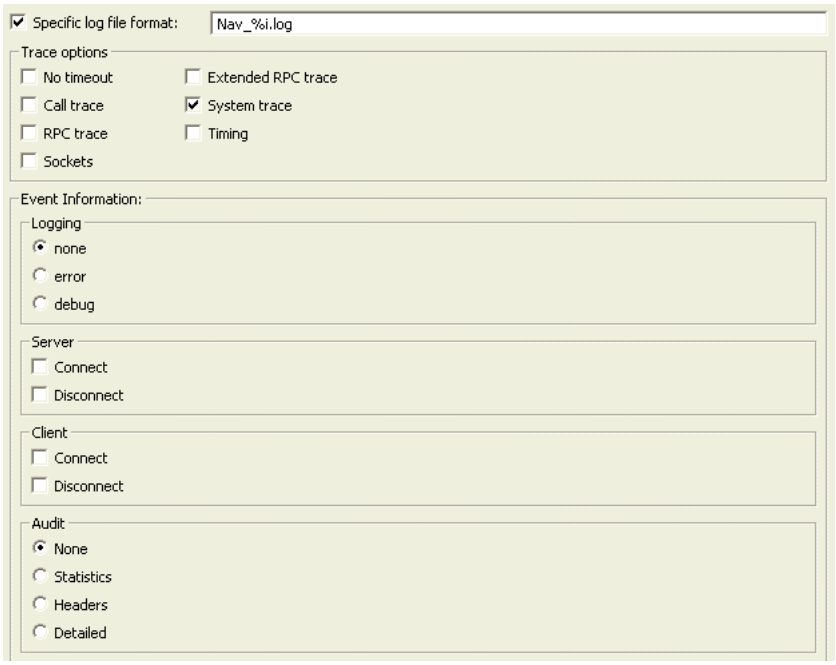
Note: You can also change daemon settings using the Configuration Explorer, by selecting a computer and scrolling the list to the required daemon. Right-click the daemon and select **Edit Daemon**.

Changes made to the daemon configuration are not implemented. They are only implemented after the configuration is reloaded using the **Reload Configuration** option in the Runtime Manager.

8. Right-click the daemon and select **End Unused Servers**. Any servers in the connection pool are closed and new servers start with the new configuration.

The **WS Logging** tab is shown in the following figure:

Figure 6–14 The WS Logging tab



The **WS Logging** tab comprises fields, as listed in the following table:

Table 6–11 WS Logging tab Components

Field	Description
Specific log file format	Defines the name and location of the server log file if you want the data written to a file instead of SYSOUT for the server process. The parameter must specify the name and the high level qualifier. The following tokens can appear in the log file template and will be replaced accordingly: ■ `%A`: workspace name ■ `%D`: date (yymmdd) ■ `%I`: instance number of the given workspace server ■ `%L`: server account's login directory ■ `%P`: server's process ID ■ `%T`: time (hhmmss) ■ `%U`: server's account name (username)
Trace options	Specifies the type of tracing to be performed. The following tracing options are available: ■ No timeout: Disables the standard RPC timeouts, setting them to a long duration (approximately an hour) to facilitate debugging. ■ Call trace: Generates a message in the server log file for each RPC function called. This is useful for troubleshooting the server. ■ RPC trace: Enables debugging messages on the server. ■ Sockets: Generates a message in the server log file for each socket operation. This is useful for troubleshooting client/server communication - providing a detailed trace of every client/server communication. ■ Extended RPC trace: Generates a verbose message in the server log file for each low-level RPC function called. This is useful for troubleshooting the server. ■ System trace: Generates operating system-specific tracing. ■ Timing: Generates a timestamp for every entry to the server log file.
Logging	Specifies the level of events that are logged for the workspace. The following event levels are available: ■ `none`: The event log only displays the IP addresses of client that have logged in and out from the workspace. ■ `error`: The event log displays the IP addresses of client that have logged in and out from the workspace as well as any errors that have been generated. ■ `debug`: The event log displays the IP addresses of client that have logged in and out from the workspace as well as any errors that have been generated and all trace results that were specified in the **Daemon Logging** tab.

Table 6–11 (Cont.) WS Logging tab Components

Field	Description
Server	Specifies the server connection events to log. The following server events are available: ■ Connect: The event log displays the server connection events. ■ Disconnect: The event log displays the server disconnect events.
Client	Specifies the type of tracing performed. The following client events are available: ■ Connect: The event log displays the client connection events. ■ Disconnect: The event log displays the client disconnect events.
Audit	This group is not applicable for use with OracleAS Adapter for Tuxedo.

WS Security

Using the **WS Security** tab, you specify the level of security at the workspace level, as opposed to the daemon level, which is set in the **Daemon Security** tab.

> **See Also:** "Daemon Security" on page C-5 for details about security.

The **WS Security** tab is used to:

- Grant administration rights for the workspace
- Determine access to the workspace by a client

The **WS Security** tab is accessed as follows:

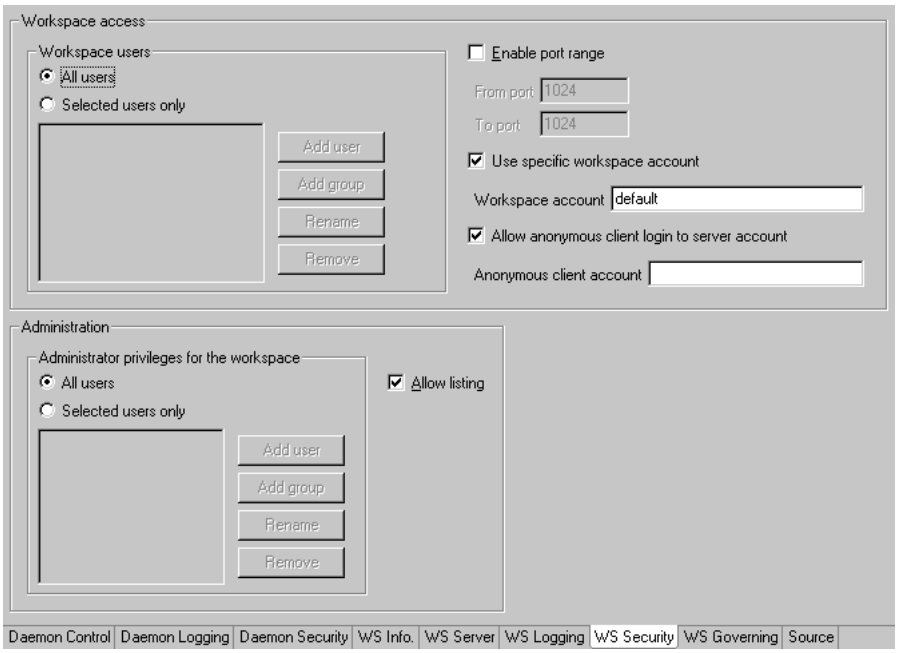
1. From the **Start** menu, select **Programs, Oracle, and then select Studio**.
2. Select the required computer from the Configuration Explorer.
3. Right-click the computer and select **Open Runtime Perspective**.
4. Expand the **Daemons** node to display the workspaces in the Runtime Explorer.
5. Right-click the workspace and select **Edit Workspace Configuration**.
6. Click the **WS Security** tab.
7. After making changes to the workspace, right-click the daemon and select **Reload Configuration**.

> **Note:** You can also change daemon settings using the Configuration Explorer, by selecting a computer and scrolling the list to the required daemon. Right-click the daemon and select **Edit Daemon**.
>
> Changes made to the daemon configuration are not implemented. They are only implemented after the configuration is reloaded using the **Reload Configuration** option in the Runtime Manager.

8. Right-click the daemon and select **End Unused Servers**. Any servers in the connection pool are closed and new servers start with the new configuration.

The **WS Security** tab is shown in the following figure:

Figure 6–15 The WS Security tab



The **WS Security** tab comprises fields, as listed in the following table:

Table 6–12 WS Security tab Components

Field	Description
Administration	Defines the users (accounts) allowed to perform administrative tasks (tasks that require administrative login) on this workspace.
Administrator privileges	Identifies the users (accounts) with administrator privileges. The following options are available: ■ All users: Indicates that anyone can access the workspace and change the settings. ■ Selected users only: The names of users (accounts) and groups that can be administrators. **Note**: If a user is not specified here, the user specified in the Workspace users area will have administrator rights for this workspace. In this case, if all users are selected for the Workspace users area, then all users have administrator rights for this workspace. ■ Allow Listing: Determines whether this workspace appears in the list of workspaces.
Workspace account	Defines the users (accounts) allowed to access the workspace, firewall access ports, workspace account, and anonymous login permissions.

Table 6–12 (Cont.) WS Security tab Components

Field	Description
Workspace users	Lists the users who are allowed to use the workspace. ■ All users: Indicates that any user who has logged on to the daemon can use the workspace. ■ Selected users only: Specifies users (accounts) and groups that can use the workspace. **Note**: If a user is not specified, any user who has logged on to the daemon can use the workspace. ■ Enable ports range: Defines the firewall ports through which you access the workspace. Specifies the range of ports available for this workspace when starting server processes. Use this option when you want to control the port number, so that Oracle Connect can be accessed through a firewall. ■ Use specific workspace account: Defines the operating system account used for the workspace. If not specified, the account name that was provided by the client is used. ■ Allow anonymous client login to server account: Defines whether this workspace can be accessed without authentication (user name/password). If anonymous login is allowed, specify the server account name to use. If this field is not specified, then the value in the Workspace account field is used.

D

Globalization Settings

OracleAS Adapters for Tuxedo provides the globalization support for the following languages:

- Arabic
- English (the default)
- French
- German
- Greek
- Hebrew
- Italian
- Japanese
- Korean
- Portuguses
- Simple Chinese
- Spanish
- Traditional Chinese
- Turkish

This appendix describes how to define the language support.

Defining the Language and Codepage

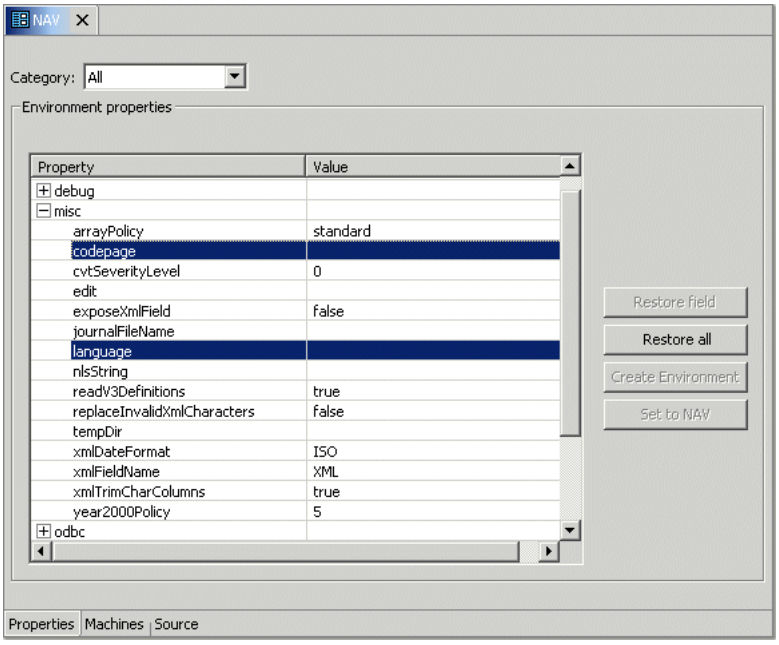
The language and codepage parameters are accessed from the computer where Oracle Studio is installed.

Perform the following steps to define the required language and codepage:

1. From the **Start** menu, select **Programs, Oracle**, and then select **Studio**.
2. Select and expand the required computer node from the Configuration Explorer.
3. Expand the Binding node.
4. Right-click **NAV** and select **Edit Binding**.
5. In the binding **Properties** tab, expand the **misc** node.

   The NAV binding **Properties** tab is shown in the following figure:

Figure 6–16 The Properties tab



6. Enter a valid value for the language. See NLS Language Codes.
7. Optionally, in the **codepage** field, specify the codepage required.

   You can skip this step, and specify a language (see the previous step). In this case, a default codepage is used.

The following table lists the codepages:

Table 6–13 NLS Language Codes

Language Name	Language Code	ASCII Platforms (Default)	EBCDIC Platforms (Default)	Alternative Codepages (EBCDIC based unless noted otherwise)
Arabic	ARA	ISO-8859-6	IBM420	Windows-1256 (ASCII based)
Chinese - Simplified	SCHI	GB2312	IBM935	-
Chinese - Traditional	TCHI	BIG5	IBM937	-
English UK	ENUK	ISO-8859-15	IBM1146	IBM037, IBM500, IBM1140, IBM1148
				ISO-8859-1 (ASCII based)
English US	ENUS	ISO-8859-15	IBM1140	IBM500, IBM1148
				ISO-8859-1 (ASCII based)
French	FRE	ISO-8859-15	IBM1147	IBM037, IBM500, IBM1140, IBM1148
				ISO-8859-1 (ASCII based)
German	GER	ISO-8859-15	IBM1141	IBM037, IBM500, IBM1140, IBM1148
				ISO-8859-1 (ASCII based)
Greek	GRK	ISO-8859-7	IBM875	-

Table 6–13 (Cont.) NLS Language Codes

Language Name	Language Code	ASCII Platforms (Default)	EBCDIC Platforms (Default)	Alternative Codepages (EBCDIC based unless noted otherwise)
Hebrew	HEB	ISO-8859-8	IBM424	-
Italian	ITL	ISO-8859-15	IBM1144	IBM037, IBM500, IBM1140, IBM1148
				ISO-8859-1 (ASCII based)
Japanese	JPN	SJIS	IBM939	-
		EUC (Solaris)		
		VMS-JP (VMS)		
Korean	KOR	KSC5601	IBM933	MS949, EUC-KR
		MS949 (Win)		(both ASCII based)
Latin International	LAT	ISO-8859-15	IBM1148	IBM037, IBM1140
				ISO-8859-1 (ASCII based)
Portuguese	POR	ISO-8859-15	IBM1140	IBM500, IBM1148
				ISO-8859-1 (ASCII based)
Russian	RUS	ISO-8859-5	IBM1154	-
Spanish	SPA	ISO-8859-15	IBM1145	IBM037, IBM500, IBM1140, IBM1148
				ISO-8859-1 (ASCII based)
Turkish	TUR	ISO-8859-9	IBM1155	IBM857

Index

Y