Oracle® Application Server Integration InterConnect

Adapter for HTTP Installation and User's Guide

10*g* Release 2 (10.1.2)

Part No. B14074-01

November 2004

ORACLE®

Oracle Application Server Integration InterConnect Adapter for HTTP Installation and User's Guide 10*g* Release 2 (10.1.2)

Part No. B14074-01

Primary Author: Vimmy K Raj, Pradeep Vasudev

Contributor: Sandeep Jain, Maneesh Joshi, Rahul Pathak, Harish Sriramulu

Contents

Send Us Your Comments

Oracle Application Server Integration InterConnect Adapter for HTTP Installation and User's Guide, 10*g* Release 2 (10.1.2)

Part No. B14074-01

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: appserverdocs_us@oracle.com
- FAX: 650-506-7375 Attn: Oracle Application Server Documentation Manager
- Postal service:

  Oracle Corporation
  Oracle Application Server Documentation Manager
  500 Oracle Parkway, M/S 1op6
  Redwood Shores, CA 94065
  USA

If you would like a reply, please give your name, address, telephone number, and electronic mail address (optional).

If you have problems with the software, please contact your local Oracle Support Services.

Preface

This Preface contains these topics:

Audience

Oracle Application Server Integration InterConnect Adapter for HTTP Installation and User's Guide is intended for system administrators of OracleAS Integration InterConnect who perform the following tasks:

- install applications
- maintain applications

To use this document, you need to know how to install and configure OracleAS Integration InterConnect.

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. 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 additional information, visit the Oracle Accessibility Program Web site at

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

Accessibility of Code Examples in Documentation

JAWS, a Windows screen reader, 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, JAWS 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.

Structure

This document contains:

Chapter 1, "Introduction"

This chapter describes the OracleAS Integration InterConnect Adapter for HTTP (HTTP adapter), and the hardware and software requirements.

Chapter 2, "Installation and Configuration"

This chapter describes installation and configuration of the HTTP adapter.

Chapter 3, "Design Time and Runtime Concepts"

This chapter describes the design time and runtime concepts of the HTTP adapter.

Appendix A, "Frequently Asked Questions"

This chapter provides answers to frequently asked questions about the HTTP adapter.

Appendix B, "Example of the adapter.ini File"

This appendix shows an example of the `adapter.ini` file.

Related Documents

For more information, refer to these Oracle resources:

- *Oracle Application Server Integration InterConnect User's Guide*
- *Oracle Application Server Integration InterConnect Installation Guide*

Printed documentation is available for sale in the Oracle Store at

`http://oraclestore.oracle.com/`

To download free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

`http://www.oracle.com/technology/membership/`

If you already have a user name and password for OTN, then you can go directly to the documentation section of the OTN Web site at

`http://www.oracle.com/technology/documentation/`

Conventions

This section describes the conventions used in the text and code examples of this documentation set. It describes:

- Conventions in Text

Conventions in Text

We use the following conventions in text to help you more quickly identify special terms. The table also provides examples of their use.

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an **index-organized table**.
Italics	Italic typeface indicates book titles or emphasis.	*Oracle Database 10g Concepts* Ensure that the recovery catalog and target database do *not* reside on the same disk.
`UPPERCASE monospace (fixed-width) font`	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, Recovery Manager keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, user names, and roles.	You can specify this clause only for a `NUMBER` column. You can back up the database by using the `BACKUP` command. Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view. Use the `DBMS_STATS.GENERATE_STATS` procedure.
`lowercase monospace (fixed-width) font`	Lowercase monospace typeface indicates executable programs, filenames, directory names, and sample user-supplied elements. *Note:* Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter `sqlplus` to start SQL*Plus. The password is specified in the `orapwd` file. Back up the datafiles and control files in the `/disk1/oracle/dbs` directory. The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table. Connect as `oe` user. The `JRepUtil` class implements these methods.
`lowercase italic monospace (fixed-width) font`	Lowercase italic monospace font represents placeholders or variables.	You can specify the *`parallel_clause`*. Run *`old_release`*`.SQL` where *`old_release`* refers to the release you installed prior to upgrading.

Conventions in Code Examples

Code examples illustrate SQL, PL/SQL, SQL*Plus, or other command-line statements. They are displayed in a monospace (fixed-width) font and separated from normal text as shown in this example:

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

The following table describes typographic conventions used in code examples and provides examples of their use.

Convention	Meaning	Example
`[ ]`	Anything enclosed in brackets is optional.	`DECIMAL (`*`digits`*` [ , `*`precision`*` ])`
`{ }`	Braces are used for grouping items.	`{ENABLE	DISABLE}`

Convention	Meaning	Example
|	A vertical bar represents a choice of two options.	{ENABLE | DISABLE} [COMPRESS | NOCOMPRESS]
...	Ellipsis points mean repetition in syntax descriptions.	CREATE TABLE ... AS *subquery*;
	In addition, ellipsis points can mean an omission in code examples or text.	SELECT *col1*, *col2*, ... , *coln* FROM employees;
Other symbols	You must use symbols other than brackets ([ ]), braces ({ }), vertical bars (|), and ellipsis points (...) exactly as shown.	acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;
Italics	Italicized text indicates placeholders or variables for which you must supply particular values.	CONNECT SYSTEM/*system_password* DB_NAME = *database_name*
UPPERCASE	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. Because these terms are not case sensitive, you can use them in either UPPERCASE or lowercase.	SELECT last_name, employee_id FROM employees; SELECT * FROM USER_TABLES; DROP TABLE hr.employees;
lowercase	Lowercase typeface indicates user-defined programmatic elements, such as names of tables, columns, or files. **Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	SELECT last_name, employee_id FROM employees; sqlplus hr/hr CREATE USER mjones IDENTIFIED BY ty3MU9;

Conventions for Windows Operating Systems

The following table describes conventions for Windows operating systems and provides examples of their use.

Convention	Meaning	Example
Click **Start**, and then choose the ***menu item***	How to start a program.	To start the Database Configuration Assistant, click **Start**, and choose **Programs**. In the Programs menu, choose **Oracle** - ***HOME_NAME*** and then click **Configuration and Migration Tools**. Choose **Database Configuration Assistant**.
File and directory names	File and directory names are not case sensitive. The following special characters are not allowed: left angle bracket (<), right angle bracket (>), colon (:), double quotation marks ("), slash (/), pipe (|), and dash (-). The special character backslash (\) is treated as an element separator, even when it appears in quotes. If the filename begins with \\, then Windows assumes it uses the Universal Naming Convention.	c:\winnt"\"system32 is the same as C:\WINNT\SYSTEM32

Convention	Meaning	Example
`C:\>`	Represents the Windows command prompt of the current hard disk drive. The escape character in a command prompt is the caret (^). Your prompt reflects the subdirectory in which you are working. Referred to as the *command prompt* in this manual.	`C:\oracle\oradata>`
Special characters	The backslash (\) special character is sometimes required as an escape character for the double quotation mark (") special character at the Windows command prompt. Parentheses and the single quotation mark (') do not require an escape character. Refer to your Windows operating system documentation for more information on escape and special characters.	`C:\>exp HR/HR TABLES=employees QUERY=\"WHERE job_id='SA_REP' and salary<8000\"`
HOME_NAME	Represents the Oracle home name. The home name can be up to 16 alphanumeric characters. The only special character allowed in the home name is the underscore.	`C:\> net start Oracle`*HOME_NAME*`TNSListener`
ORACLE_HOME and *ORACLE_BASE*	In releases prior to Oracle8*i* release 8.1.3, when you installed Oracle components, all subdirectories were located under a top level *ORACLE_HOME* directory. This release complies with Optimal Flexible Architecture (OFA) guidelines. All subdirectories are not under a top level *ORACLE_HOME* directory. There is a top level directory called *ORACLE_BASE* that by default is `C:\oracle\product\10.1.0`. If you install the latest Oracle release on a computer with no other Oracle software installed, then the default setting for the first Oracle home directory is `C:\oracle\product\10.1.0\db_`*n*, where *n* is the latest Oracle home number. The Oracle home directory is located directly under *ORACLE_BASE*. All directory path examples in this guide follow OFA conventions. Refer to *Oracle Database Installation Guide for Windows* for additional information about OFA compliances and for information about installing Oracle products in non-OFA compliant directories.	Change to the *ORACLE_BASE*`\`*ORACLE_HOME*`\rdbms\admin` directory.

1 Introduction

This chapter provides an overview on how to use Oracle Application Server Integration InterConnect (OracleAS Integration InterConnect) Adapter for Hypertext Transfer Protocol (HTTP adapter). This guide provides information about installing and configuring the HTTP adapter to use either Secure Socket Layer (SSL) functionality (HTTPS) or non-SSL functionality (HTTP).

This chapter contains the following topics:

1.1 HTTP Adapter Overview

The HTTP adapter enables an HTTP application to be integrated with other applications using OracleAS Integration InterConnect. The HTTP adapter is useful in all Enterprise Application Integration (EAI) environments that use HTTP. EAI is the integration of applications and business processes within the same company.

The HTTP adapter can monitor incoming HTTP requests received by the HTTP adapter servlet. The HTTP adapter is also capable of sending messages to remote Web servers by proxy host. The payload for this adapter can be XML data or D3L data.

Figure 1–1 depicts the data flow of incoming messages from an HTTP client to OracleAS Integration InterConnect. Incoming messages are sent to a servlet provided by the HTTP adapter. The servlet sends the message to an HTTP receiver in the adapter through Remote Method Invocation (RMI).

Figure 1–1 Incoming Messages

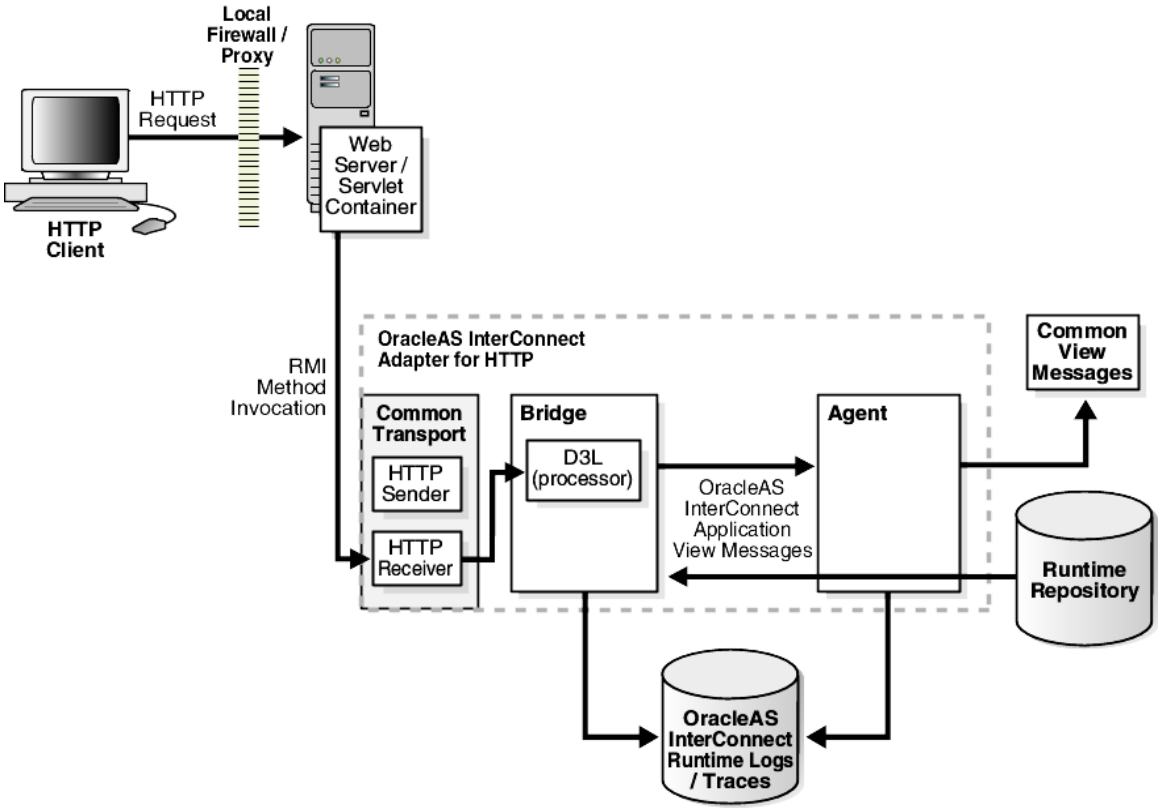
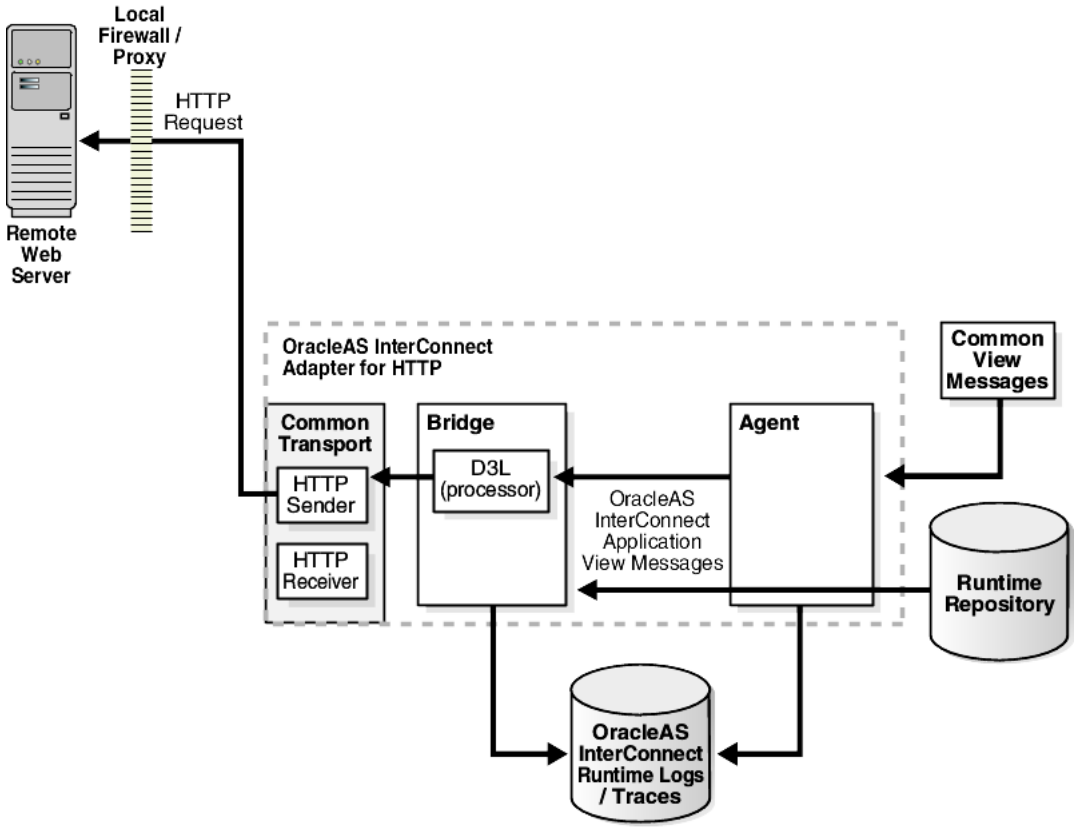


Figure 1–2 depicts the data flow of outgoing messages from OracleAS Integration InterConnect to a remote Web server. Outgoing messages are sent from an HTTP sender object in the adapter to the remote Web server.

Figure 1–2 Outgoing Messages



See Also: *Oracle Application Server Security Guide* for additional information about SSL and Oracle Wallet Manager

1.2 HTTP Adapter System Requirements

The following sections describe the HTTP adapter system requirements:

- Hardware Requirements
- Software Requirements

1.2.1 Hardware Requirements

Table 1–1 lists the hardware requirements for the computer where the HTTP adapter will be installed.

Table 1–1 Hardware Requirements

Hardware	Windows	UNIX
Disk Space	500 MB	500 MB
Memory	128 MB	128 MB

1.2.2 Software Requirements

The following sections describe the HTTP adapter software requirements:

- Operating System Requirements

- JRE Requirements
- Servlet Requirements

1.2.2.1 Operating System Requirements

Table 1–2 lists the operating system requirements for the computer where the HTTP adapter will be installed.

Table 1–2 Operating System Requirements

Operating System	Version
HP Tru64	HP Tru64 UNIX (Alpha) 5.1b
HP-UX	HP-UX (PA-RISC) 11.11, 11.23
IBM AIX	AIX (POWER) version 5.2
Linux (x86)	Red Hat Enterprise Linux 2.1, 3.0 SuSE SLES8, SLES9
Sun SPARC Solaris	Sun SPARC Solaris 2.8 and 2.9
Microsoft Windows	Windows XP Professional, Windows 2000(SP3 or higher)

1.2.2.2 JRE Requirements

OracleAS Integration InterConnect uses Java Runtime Environment (JRE) 1.4, which is installed with its components.

1.2.2.3 Servlet Requirements

The HTTP adapter requires Oracle Application Server Containers for J2EE (OC4J), which is provided by Oracle Application Server. OC4J is not required to be installed on the computer where the HTTP adapter is installed.

> **See Also:** *Oracle Application Server Containers for J2EE User's Guide* for additional information on OC4J

1.3 HTTP Adapter Features

The HTTP adapter has the following features:

- Supports HTTP versions 1.0 and 1.1.
- Provides SSL functionality (known as HTTPS) with Oracle JavaSSL, which uses a wallet generated from Oracle Wallet Manager.
- Allows selection of suitable cipher suites for an SSL connection. Cipher suites control the combination of encryption and data integrity used by SSL.
- Supports sending HTTP requests and receiving HTTP replies through a proxy server.
- Supports the synchronous and asynchronous request/reply and publish/subscribe messaging paradigms.
- supports Microsoft Internet Information Server (IIS) through use of the OracleAS Proxy Plugin for Microsoft IIS for inbound communications.

1.4 Known HTTP Adapter Limitations

The HTTP adapter has the following limitations:

- The point-to-point messaging functionality is currently not supported.
- All the incoming messages for an HTTP adapter application are received through a single HTTP adapter servlet.
- The sending and receiving applications must support HTTP.
- Only the HTTP `POST` access method is supported by the HTTP adapter servlet to receive incoming messages.

2

Installation and Configuration

This chapter describes how to install and configure the HTTP adapter. It contains the following topics:

- Installing the HTTP Adapter
- Configuring the HTTP Adapter

2.1 Installing the HTTP Adapter

The HTTP adapter must be installed in an existing Oracle home Middle Tier for OracleAS Integration InterConnect 10*g* Release 2 (10.1.2).

This section contains the following topics:

- Preinstallation Tasks
- Installation Tasks
- Postinstallation Tasks

2.1.1 Preinstallation Tasks

Consult the following guides before installing the HTTP adapter:

- *Oracle Application Server Installation Guide* for information about Oracle Universal Installer startup.
- *Oracle Application Server Integration InterConnect Installation Guide* for information on mounting CD-ROMs, software, hardware, and system requirements for OracleAS Integration InterConnect.

> **Note:** OracleAS Integration InterConnect Hub is installable through the OracleAS Integration InterConnect Hub installation type. You must install the OracleAS Integration InterConnect Hub before proceeding with the HTTP adapter installation.

2.1.2 Installation Tasks

To install the HTTP adapter:

1. In the Available Product Components page of the OracleAS Integration InterConnect installation, select **HTTP adapter**, and click **Next**.
2. The Set Oracle Wallet Password screen is displayed. Enter and confirm the password on the screen, which will be used to administer OracleAS Integration InterConnect installation. Click **Next**.

- Go to step 3, if installing the HTTP adapter in an OracleAS Middle Tier Oracle home that does not have an InterConnect component already installed. Ensure that the OracleAS Integration InterConnect hub has been installed.
- Go to step 4, if installing the HTTP adapter in an OracleAS Middle Tier Oracle home that has an existing InterConnect component. Ensure that it is a home directory to an OracleAS Integration InterConnect component.

3. The Specify Hub Database Connection page is displayed. Enter information in the following fields:
   - Host Name: The host name of the computer where the hub database is installed.
   - Port Number: The TNS listener port for the hub database.
   - Database SID: The System Identifier (SID) for the hub database.
   - Password: The password for the hub database user.
4. Click **Next**. The Specify HTTP Adapter Name page is displayed.
5. Enter the application to be defined. Blank spaces are not permitted. The default value is `myHTTPApp`.
6. Click **Next**. The Specify HTTP Adapter Usage screen is displayed.
7. Select one of the options and go to the specified step.

If You Select...	Then Click Next and Go to Step...
Configure for both sending and receiving messages	8
Configure for sending messages ONLY	8
Configure for receiving messages ONLY	10

Note: You can change the values for these selections later by editing the parameter settings in the `adapter.ini` file.

8. Enter the URL `http://hostname:port/path/` in the Configure Sending Endpoint Information page. The URL is used by the HTTP adapter for sending messages.
9. Click **Next**. The installation screen that appears is based on the selection made in Step 7.

If You Selected...	Then Go to Step...
Configure for both sending and receiving messages	10
Configure for sending messages ONLY	12

10. Enter the following information in the Configure Receiving Endpoint Information page:
    - Hostname: The hostname of the HTTP server from which OracleAS Integration InterConnect receives messages.
    - Port Number: The port number of the HTTP server.

11. Click **Next**. The Summary page is displayed.
12. Click **Install** to install the HTTP adapter. The adapter is installed in the following directory, depending on the operating system:

Platform	Directory
UNIX	*ORACLE_HOME*/integration/interconnect/adapters/*Application*
Windows	*ORACLE_HOME*\integration\interconnect\adapters*Application*

You defined the value of `Application` in Step 5. A `webapps` subdirectory is created in the *`Application`* directory, which includes the following files for the HTTP application:

- An EAR file (`oai.ear`)
- A `web.xml` file located in the `WEB-INF` directory
- An `application.xml` file located in the `META-INF` directory

13. Click **Exit** on the Installation page to exit the HTTP adapter installation.

2.1.3 Postinstallation Tasks

HTTP adapter installation creates the `adapter.ini` file that consists of configuration parameters read by the HTTP adapter at startup. These configuration parameter settings are suitable for most HTTP application environments. To customize the `adapter.ini` file parameter settings for the HTTP application, refer to the following sections:

- Customizing the Payload Type
- Customizing the Sending Endpoints
- Customizing the Authentication Scheme
- Customizing a Proxy Host
- Customizing a Secure Socket Layer Environment
- Customizing the Receiving Endpoints
- Deploying an EAR File Manually

See Also: Table 2–1 on page 2-7 for the location of the adapter on different platforms

2.1.3.1 Customizing the Payload Type

Payload data is the data sent between applications. To change the payload type from the default of XML to D3L, edit the parameters in the `adapter.ini` file.

1. Set the `ota.type` parameter to the payload type D3L.

```
ota.type=D3L
```

2. Copy the D3L XML files associated with the HTTP application to the directory in which the `adapter.ini` file is located.

3. Set the `ota.d3ls` parameter to specify the D3L files associated with the HTTP application. For example:

```
ota.d3ls=person1.xml,person2.xml
```

See Also: `ota.type` and `ota.d3ls` parameter descriptions in Table 2–9 for additional information

2.1.3.2 Customizing the Sending Endpoints

To customize the behavior of the sending endpoints (destinations) for messages, edit the parameters in the `adapter.ini` file.

To customize the sending endpoints, set the http.sender.timeout parameter to the desired timeout interval in milliseconds. This parameter automatically defaults to a value of 60000 during installation. For example:

```
http.sender.timeout=10000
```

See Also: The `http.sender.timeout` parameter description in Table 2–9 for additional information

2.1.3.3 Customizing the Authentication Scheme

To use a custom authentication scheme, edit the parameters in the `adapter.ini` file. These parameters are not automatically set to default values during installation.

To customize the authentication scheme:

1. Set the `http.sender.authtype` parameter to the authentication type to use. For example:

```
http.sender.authtype=basic
```

2. Set the `http.sender.realm` parameter to the realm for the authentication scheme. For example:

```
http.sender.realm=ipt
```

3. Set the `http.sender.username` parameter to the authentication user name. For example:

```
http.sender.username=joe
```

4. Set the `http.sender.password` parameter to the authentication password. For example:

```
http.sender.password=100100101
```

See Also:

- Table 2–9, " HTTP Adapter-specific Parameters"
- "How do I secure my passwords?" for instructions on how to modify and retrieve the password

2.1.3.4 Customizing a Proxy Host

To use a proxy host, edit the parameters in the `adapter.ini` file. These parameters are not set to default values during installation.

To customize a proxy host:

1. Set the `http.sender.proxy_host` parameter to the hostname of the proxy server. For example:

   ```
   http.sender.proxy_host=www-proxy.foo.com
   ```

2. Set the `http.sender.proxy_port` parameter to the port number of the proxy server. For example:

   ```
   http.sender.proxy_port=80
   ```

See Also: Table 2–9, " HTTP Adapter-specific Parameters"

2.1.3.5 Customizing a Secure Socket Layer Environment

To send messages using the Secure Socket Layer (SSL) environment, edit the following parameters in the `adapter.ini` file. These parameters are not set to default values during installation.

To customize an SSL environment:

1. Set the `http.sender.wallet_location` parameter to the directory path and name of the wallet file. For example:

   ```
   http.sender.wallet_location=/private/foo/certdb.txt
   ```

   `certdb.txt` is the name of the flat file exported from the Oracle Wallet manager. In the `http.sender.wallet_location` parameter, you may need to use Oracle Wallet Manager to add additional trusted certificates from the HTTP server to avoid incomplete certificate chain error.

2. Set the `http.sender.wallet_password` parameter to the Oracle Wallet Manager password. For example:

   ```
   http.sender.wallet_password=4341193845566
   ```

Note: All passwords are stored in Oracle Wallet. Refer to "How do I secure my passwords?" for more details on how to modify and retrieve the password using Oracle Wallet.

3. Set the `http.sender.cipher_suites` parameter to the cipher suites used in the secure connection. For example:

   ```
   http.sender.cipher_suites=SSL_RSA_WITH_NULL_SHA,SSL_RSA_WITH_3DES_EDE_CBC_SHA
   ```

See Also: The following parameter descriptions for additional information:

- Table 2–9, " HTTP Adapter-specific Parameters"
- "How do I secure my passwords?" for instructions on how to modify and retrieve the password

2.1.3.6 Customizing the Receiving Endpoints

To customize the behavior of messages in the receiving application, edit the parameter in the `adapter.ini` file.

Set the `http.receiver.registry_port` parameter to the RMI registry port for communicating with the servlet. This parameter automatically defaults to a value of `9901` during installation. For example:

```
http.receiver.registry_port=3500
```

See Also: Table 2–9, " HTTP Adapter-specific Parameters"

2.1.3.7 Deploying an EAR File Manually

If OC4J is installed on a separate computer from the HTTP adapter, then manually edit the `web.xml` file and deploy the EAR file (`oai.ear`) located in the directory of the HTTP adapter.

To manually deploy an EAR file:

1. Change to the directory where the HTTP application is installed. For example,

   ```
   cd myHTTPapp
   ```

   myHTTPapp is the value defined in Step 4 on page 2-2.

2. Extract all the files from the `oai.ear` file:

   ```
   jar xvf oai.ear
   ```

3. Extract all the files from the `oai.war` file:

   ```
   jar xvf oai.war
   ```

4. Change to the WEB-INF directory:

   ```
   cd WEB-INF
   ```

5. Edit the `web.xml` file.

   The `web.xml` file specifies the RMI information. This must match the settings in the `adapter.ini` file.

   The following `web.xml` file shows the `rmiHost` parameter with a computer hostname setting of `prodserver10`:

   ```
   <init-param>
     <param-name>rmiHost</param-name>
     <param-value>prodserver10</param-value>
   </init-param>
   <init-param>
     <param-name>rmiPort</param-name>
     <param-value>9901</param-value>
   </init-param>
   <init-param>
     <param-name>instanceName</param-name>
     <param-value>oai</param-value>
   </init-param>
   <!-- set the following parameters if logging is needed. -->
   <init-param>
     <param-name>isLogOn</param-name>
     <!-- enter true/false -->
     <param-value>false</param-value>
   </init-param>
   <init-param>
     <param-name>logDir</param-name>
     <!-- directory where log file is placed. -->
   ```

```
  <param-value></param-value>
</init-param>
<init-param>
  <param-name>logLevel</param-name>
  <!-- choose one of the levels: debug, status, or  error -->
  <param-value></param-value>
 </init-param>
```

> **Note:** The `rmiHost` parameter must match the hostname of the computer where the HTTP adapter is installed. The HTTP adapter functions as the RMI server. The transport servlet makes an RMI call to submit the requests sent by the external application. You can also edit the logging options that are turned off by default.

6. Change to the directory where the HTTP application is installed:

   ```
   cd myHttpApp
   ```

7. Restore the `oai.war` and `oai.ear` files:

   ```
   jar cvf oai.war WEB-INF
   jar cvf oai.ear META-INF/ oai.war
   ```

8. Deploy the EAR file.

   > **See Also:** *Oracle Application Server Administrator's Guide* for instructions on using the Distributed Configuration Management (DCM) command line utility to deploy the EAR file

2.2 Configuring the HTTP Adapter

After an HTTP adapter installation, you can configure it for your needs. The following tables describe the location and details of the configuration files.

Table 2–1 describes the location where the adapter is installed.

Table 2–1 HTTP Adapter Directory

Platform	Directory
UNIX	*ORACLE_HOME*`/integration/interconnect/adapters/Application`
Windows	*ORACLE_HOME*`\integration\interconnect\adapters\Application`

Table 2–2 describes the various executable files of the HTTP adapter.

Table 2–2 HTTP Executable Files

File	Description
`start` (UNIX)	Does not use parameters, starts the adapter.
`start.bat` (Windows)	Does not use parameters, starts the adapter.
`stop` (UNIX)	Does not use parameters, stops the adapter.
`stop.bat` (Windows)	Does not use parameters, stops the adapter.

Table 2–3 describes the HTTP adapter configuration files.

Table 2–3 HTTP Configuration Files

File	Description
`adapter.ini` (UNIX)	Consists of all the initialization parameters that the adapter reads at startup.
`adapter.ini` (Windows)	Consists of all the initialization parameters that the adapter reads at startup.

Table 2–4 describes the directories used by the HTTP adapter.

Table 2–4 HTTP Directories

Directory	Description
`logs`	The adapter activity is logged in subdirectories of the logs directory. Each time the adapter is run, a new subdirectory is created for the oailog.txt log file.
`persistence`	The messages are made available in this directory. Do not edit this directory or its files.

2.2.1 Ini File Settings

The following are the `.ini` files used to configure the HTTP adapter:

- hub.ini Files
- adapter.ini Files

2.2.1.1 hub.ini Files

The HTTP adapter connects to the hub database using the parameters in the `hub.ini` file located in the `hub` directory. Table 2–5 lists the parameter names, descriptions for each parameter, and examples.

Table 2–5 hub.ini Parameters

Parameter	Description	Example
hub_host	The name of the computer hosting the hub database. There is no default value. The value is set during installation.	`hub_host=mpscottpc`
hub_instance	The SID of the hub database. There is no default value. The value is set during installation.	`hub_instance=orcl`
hub_port	The TNS listener port number for the hub database instance. There is no default value. The value is set during installation.	`hub_port=1521`
hub_username	The name of the hub database schema (or user name). The default value is `ichub`.	`hub_username=ichub`
repository_name	The name of the repository that communicates with the adapter. The default value is `InterConnectRepository`.	`repository_name=InterConnectRepository`

Oracle Real Application Clusters hub.ini Parameters

When a hub is installed on an Oracle Real Application Clusters database, the parameters listed in Table 2–6 represent information about additional nodes used for connection and configuration. These parameters are in addition to the default parameters for the primary node. In Table 2–6, `x` represents the node number. The

number is between 2 and the number of nodes. For example, if the cluster setup contains 4 nodes, `x` can be a value between 2 and 4.

Table 2–6 Oracle Real Application Clusters Hub.ini Parameters

Parameter	Description	Example
hub_host*x*	The host where the Real Application Clusters database is installed.	`hub_host2=dscott13`
hub_instancex	The instance on the respective node.	`hub_instance2=orcl2`
hub_num_nodes	The number of nodes in a cluster.	`hub_num_nodes=4`
hub_port*x*	The port where the TNS listener is listening.	`hub_port2=1521`

2.2.1.2 adapter.ini Files

The agent component of the HTTP adapter reads the `adapter.ini` file at runtime to access information on configuring the HTTP adapter parameter. Table 2–7 lists the parameter names, descriptions for each parameter, and examples.

Table 2–7 adapter.ini Parameters

Parameter	Description	Example
agent_admin_port	Specifies the port through which the adapter can be accessed through firewalls. Possible Value: A valid port number. Default Value: None.	`agent_admin_port=1059`
agent_delete_file_cache_at_startup	Specifies whether to delete the cached metadata during startup. If any agent caching method is enabled, then metadata from the repository is cached locally on the file system. Set the parameter to `true` to delete all cached metadata on startup. Possible Values: `true` or `false`. Default Value: `false`. **Note:** After changing metadata or DVM tables for the adapter in iStudio, you must delete the cache to guarantee access to new metadata or table information.	`agent_delete_file_cache_at_startup=false`
agent_dvm_table_caching	Specifies the Domain Value Mapping (DVM) table caching algorithm. Possible values: ▪ `startup`: Cache all DVM tables at startup. This may take a while if there are a lot of tables in the repository. ▪ `demand`: Cache tables as they are used. ▪ `none`: No caching. This slows down performance. Default Value: `demand`.	`agent_dvm_table_caching=demand`
agent_log_level	Specifies the amount of logging necessary. Possible values: `0`=errors only `1`=status and errors `2`=trace, status, and errors Default Value: 1.	`agent_log_level=2`

Table 2–7 (Cont.) adapter.ini Parameters

Parameter	Description	Example
agent_lookup_table_caching	Specifies the lookup table caching algorithm. Possible values: ■ `startup`: Cache all lookup tables at startup. This may take a while if there are a lot of tables in the repository. ■ `demand`: Cache tables as they are used. ■ `none`: No caching. This slows down performance. Default Value: `demand`.	`agent_lookup_table_caching=demand`
agent_max_ao_cache_size	Specifies the maximum number of application object metadata to cache. Possible Value: An integer greater than or equal to `1`. Default Value: `200`.	`agent_max_ao_cache_size=200`
agent_max_co_cache_size	Specifies the maximum number of common object metadata to cache. Possible Value: An integer greater than or equal to `1`. Default Value: `100`.	`agent_max_co_cache_size=100`
agent_max_dvm_table_cache_size	Specifies the maximum number of DVM tables to cache. Possible Value: An integer greater than or equal to `1`. Default Value: `200`.	`agent_max_dvm_table_cache_size=200`
agent_max_lookup_table_cache_size	Specifies the maximum number of lookup tables to cache. Possible Value: An integer greater than or equal to `1`. Default Value: `200`.	`agent_max_lookup_table_cache_size=200`
agent_max_message_metadata_cache_size	Specifies the maximum number of message metadata (publish/subscribe and invoke/implement) to cache. Possible Value: An integer greater than or equal to `1`. Default Value: `200`.	`agent_max_message_metadata_cache_size=200`
agent_max_queue_size	Specifies the maximum size internal OracleAS Integration InterConnect message queues can grow. Possible Value: An integer greater than or equal to `1`. Default Value: `1000`.	`agent_max_queue_size=1000`
agent_message_selector	Specifies conditions for message selection when the adapter registers its subscription with the hub. Possible Value: A valid Oracle Advanced Queue message selector string (like `'%,aqapp,%'`). Default Value: None.	`agent_message_selector=%,aqapp,%`
agent_metadata_caching	Specifies the metadata caching algorithm. Possible values: ■ `startup`: Cache everything at startup. This may take time if there are a lot of tables in the repository. ■ `demand`: Cache metadata as it is used. ■ `none`: No caching. This slows down performance. Default Value: `demand`.	`agent_metadata_caching=demand`

Table 2–7 (Cont.) adapter.ini Parameters

Parameter	Description	Example
agent_persistence_ cleanup_interval	Specifies how often to run the persistence cleaner thread in milliseconds. Possible Value: An integer greater than or equal to `30000` milliseconds. Default Value: `60000`.	`agent_persistence_ cleanup_ interval=60000`
agent_persistence_ queue_size	Specifies the maximum size of internal OracleAS Integration InterConnect persistence queues. Possible Value: An integer greater than or equal to `1`. Default Value: `1000`.	`agent_persistence_ queue_size=1000`
agent_persistence_ retry_interval	Specifies how often the persistence thread retries when it fails to send an OracleAS Integration InterConnect message. Possible Value: An integer greater than or equal to `5000` milliseconds. Default Value: `60000`.	`agent_persistence_ retry_interval=60000`
agent_pipeline_from_ hub	Specifies whether to turn on the pipeline for messages from the hub to the bridge. If you set the pipeline to `false`, then the file persistence is not used in that direction. Possible Value: `true`, `false` Default Value: `false`.	`agent_pipeline_from_ hub=false`
agent_pipeline_to_ hub	Specifies whether to turn on the pipeline for messages from the bridge to the hub. If you set the pipeline to `false`, then the file persistence is not used in that direction. Possible Value: `true`, `false`. Default Value: `false`.	`agent_pipeline_to_ hub=false`
agent_reply_message_ selector	Specifies the application instance to which the reply must be sent. This parameter is used only if multiple adapter instances exist for the given application and given partition. Possible Value: A string built using the application name (parameter:application) concatenated with the instance number (parameter:instance_number). Default Value: None.	If `application=httpapp, instance_number=2,` then `agent_reply_message_ selector=receipient_ list like '%,httpapp2,%'`
agent_reply_ subscriber_name	Specifies the subscriber name used when multiple adapter instances are used for the given application and given partition. This parameter is optional if only one instance is running. Possible Value: The application name (parameter:application) concatenated with the instance number (parameter:instance_number). Default Value: None.	If `application=httpapp` and `instance_number=2,` then `agent_reply_ subscriber_ name=httpapp2`
agent_subscriber_ name	Specifies the subscriber name used when this adapter registers its subscription. Possible Value: A valid Oracle Advanced Queue subscriber name. Default Value: None.	`agent_subscriber_ name=httpapp`

Table 2–7 (Cont.) adapter.ini Parameters

Parameter	Description	Example
agent_throughput_ measurement_ enabled	Specifies if the throughput measurement is enabled. Set this parameter to `true` to turn on all throughput measurements. Default Value: `true`.	`agent_throughput_ measurement_ enabled=true`
agent_tracking_ enabled	Specifies if message tracking is enabled. Set this parameter to `false` to turn off tracking of messages. Set this parameter to `true` to track messages with tracking fields set in iStudio. Default Value: `true`.	`agent_tracking_ enabled=true`
agent_use_custom_ hub_dtd	Specifies whether to use a custom DTD for the common view message when handing it to the hub. By default, adapters use a specific OracleAS Integration InterConnect DTD for all messages sent to the hub. Set this parameter to `true` to have the adapter use the DTD imported for the message of the common view instead of the OracleAS Integration InterConnect DTD. Default Value: None.	`agent_use_custom_hub_ dtd=false`
application	Specifies the name of the application to which this adapter connects. This must match with the name specified in iStudio while creating metadata. Possible Value: An alphanumeric string. Default Value: None.	`application=httpapp`
encoding	Specifies the character encoding for published messages. The adapter uses this parameter to generate encoding information for the encoding tag of transformed OracleAS Integration InterConnect messages. OracleAS Integration InterConnect represents messages internally as XML documents. Possible Value: A valid character encoding. Default Value: `UTF-8`. When there is no existing encoding in the subscribed message, this parameter will be used to explicitly specify the encoding of the published message.This parameter will be ignored when the encoding already exists in the subscribed message.	`encoding=Shift_JIS`
external_dtd_base_url	Specify the base URL for loading external enitites and DTDs.This specifies to the XML parser to resolve the external entities in the instance document using the given URL. Possible Value: A URL. Default Value: The URL of the current user directory.	external_dtd_base_ url=file://C:\InterConnect1 0_1_2\adapters\AQApp\
instance_number	Specifies the instance number to which this adapter corresponds. Specify a value only if you have multiple adapter instances for the given application with the given partition. Possible Value: An integer greater than or equal to `1`. Default Value: None.	`instance_number=1`

Table 2–7 (Cont.) adapter.ini Parameters

Parameter	Description	Example
nls_country	Specifies the ISO country code. The codes are defined by ISO-3166. Possible Value: A valid code. A full list of the codes is available at http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html Default Value: US. **Note**: This parameter specifies date format. It is applicable for the date format only.	nls_country=US
nls_date_format	Specifies the format for a date field expressed as a string. Possible Value: Any valid date format pattern as shown in Table 2–8 for the definitions of the format characters. Default Value: EEE MMM dd HHmmss zzz yyyy.	Date format pattern dd/MMM/yyyy can represent 01/01/2003. nls_date_format=dd-MMM-yy Multiple date formats can be specified as num_nls_formats=2 nls_date_format1=dd-MMM-yy nls_date_format2=dd/MMM/yy
nls_language	Specifies the ISO language code. The codes are defined by ISO-639. Possible Value: A valid code. A full list of these codes is available at http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt Default Value: en. **Note**: This parameter specifies date format. It is applicable for the date format only.	nls_language=en
partition	Specifies the partition this adapter handles as specified in iStudio. Possible Value: An alphanumeric string. Default Value: None.	partition=germany
service_class	Specifies the entry class for the Windows service. Possible Value: oracle/oai/agent/service/AgentService. Default Value: None.	service_class=oracle/oai/agent/service/AgentService
service_classpath	Specifies the class path used by the adapter JVM. If a custom adapter is developed and the adapter is to pick up any additional jar files, then add the files to the existing set of jar files. Possible Value: A valid PATH setting. Default Value: None. This parameter is for Microsoft Windows only.	service_classpath=D:\oracle\oraic\integration\interconnect\lib\oai.jar;D:\oracle\oraic\jdbc\classes12.zip

Table 2–7 (Cont.) adapter.ini Parameters

Parameter	Description	Example
service_jdk_dll	Specifies the Dynamic Link Library(DLL) that the adapter JVM should use. Possible Value: A valid `jvm.dll`. Default Value: `jvm.dll`. This parameter is for Microsoft Windows only.	`service_jdk_dll=jvm.dll`
service_jdk_version	Specifies the JDK version that the adapter Java VM should use. Possible Value: A valid JDK version number. Default Value: 1.4 This parameter is for Microsoft Windows only.	`service_jdk_version=1.4`
service_max_heap_size	Specifies the maximum heap size for the adapter JVM. Possible Value: A valid JVM heap size. Default Value: `536870912`. This parameter is for Microsoft Windows only.	`service_max_heap_size=536870912`
service_max_java_stack_size	Specifies the maximum size the JVM stack can grow. Possible Value: A valid JVM maximum stack size. Default Value: Default value for the JVM. This parameter is for Microsoft Windows only.	`service_max_java_stack_size=409600`
service_max_native_stack_size	Specifies the maximum size the JVM native stack can grow. Possible Value: The valid JVM maximum native stack size. Default Value: Default value for the JVM. This parameter is for Microsoft Windows only.	`service_max_native_size=131072`
service_min_heap_size	Specifies the minimum heap size for the adapter JVM. Possible Value: The valid JVM heap size. Default Value: `536870912`. This parameter is for Microsoft Windows only.	`service_min_heap_size=536870912`

Table 2–7 (Cont.) adapter.ini Parameters

Parameter	Description	Example
service_num_vm_args	Specifies the number of `service_vm_arg`*number* parameters specified in JVM. Possible Value: The number of `service_vm_arg`*number* parameters. Default Value: None. This parameter is for Microsoft Windows only.	`service_num_vm_args=1`
service_path	Specifies the environment variable `PATH`. The `PATH` variable is set before starting the Java Virtual Machine (JVM). Typically, list all directories that contain necessary DLLs. Possible Value: The valid `PATH` environment variable setting. Default Value: None. This parameter is for Microsoft Windows only.	`service_path=%JREHOME%\bin;D:\oracle\oraic\bin`
service_vm_arg*number*	Specifies any additional arguments to the JVM. For example, to retrieve line numbers in any of the stack traces, set `service_vm_arg1=java.compiler=NONE`. If a list of arguments exists, then use multiple parameters as shown in the example, by incrementing the last digit by `1`. Possible Value: A valid JVM arguments. Default Value: None. This parameter is for Microsoft Windows only.	`service_vm_arg1=java.compiler=NONE` `service_vm_arg2=oai.adapter=.aq`

Table 2–8 shows the reserved characters used to specify the value of the `nls_date_format` parameter. Use these characters to define date formats.

Table 2–8 Reserved Characters for the nls_date_format Parameter

Letter	Description	Example
G	Era designator	`AD`
y	Year	`1996` or `96`
M	Month in year	`July` or `Jul` or `07`
w	Week in year	`27`
W	Week in month	`2`
D	Day in year	`189`
d	Day in month	`10`
F	Day of week in month	`Number 2`
E	Day in week	`Tuesday` or `Tue`
a	a.m./p.m. marker	`P.M.`
H	Hour in day (0-23)	`0`
k	Hour in day (1-24)	`24`
K	Hour in a.m./p.m. (0-11)	`0`
h	Hour in a.m./p.m. (1-12)	`12`
m	Minute in hour	`30`

Table 2–8 (Cont.) Reserved Characters for the nls_date_format Parameter

Letter	Description	Example
s	Second in minute	`55`
S	Millisecond	`978`

HTTP Adapter-specific Parameters

Table 2–9 lists the parameters specific to the HTTP adapter.

Table 2–9 HTTP Adapter-specific Parameters

Parameter	Description	Example
bridge_class	Specifies the entry class for the HTTP adapter. Once set, the value cannot be modified. Possible Value: `oracle.oai.agent.adapter.technology.TechBridge`. Default Value: None.	`bridge_class=oracle.oai.agent.adapter.technology.TechBridge`
http.receiver.customized_class	Specifies the class name for customizing the HTTP response. Default Value: None.	`http.receiver.customized_class=MyBanner`
http.receiver.customizer_class	Specifies the class name for customizing the HTTP sender. Default Value: `oracle.oai.agent.adapter.technology.HTTPDefaultSenderCustomizer.`	`http.receiver.customizer_class=MyHTTPReceiverCustomizer`
http.receiver.instance_name	Specifies the instance name of the HTTP receiver. If the default value is not used, then the `instanceName` of the initial parameter of the transport servlet must be modified to match this instance name. Default Value: `oai`.	`http.receiver.instance_name=oai`
http.receiver.registry_port	Specifies the RMI port used by the HTTP receiver. Default Value: `9901`.	`http.receiver.registry_port=9901`
http.reqreply.mode	Specifies the type of mode, synchronous or asynchronous, that can be set for the request reply messaging paradigm. Possible values: sync, async. In the case of async, the reply will be sent to the send endpoint defined by `ota.send.endpoint`. Default Value: async.	`http.reqreply.mode=sync`
http.reqreply.syncmode.timeout	Specifies the time period the adapter should wait for a reply. Set this property only if `http.reqreply.mode` is synchronous. Set this property to customize the time period the adapter should wait for a reply. The value should be in milliseconds and -1 will be interpreted as infinite. Default Value: 60s.	`http.reqreply.syncmode.timeout=6000`

Table 2–9 (Cont.) HTTP Adapter-specific Parameters

Parameter	Description	Example
http.sender.authtype	Specifies if authentication is needed. Possible Value: `basic` or `digest`. Default Value: None.	`http.sender.authtype=basic`

http.sender.cipher_suites	Specifies the cipher to use for encrypting messages. This is an optional parameter for choosing the cipher suites. The selections are:
	`SSL_RSA_WITH_3DES_EDE_CBC_SHA`
	`SSL_RSA_WITH_RC4_128_SHA`
	`SSL_RSA_WITH_RC4_128_MD5`
	`SSL_DH_anon_WITH_3DES_EDE_CBC_SHA`
	`SSL_DH_anon_WITH_RC4_128_MD5`
	`SSL_DH_anon_WITH_DES_CBC_SHA`
	`SSL_RSA_WITH_DES_CBC_SHA`
	`SSL_RSA_EXPORT_WITH_RC4_40_MD5`
	`SSL_RSA_EXPORT_WITH_DES40_CBC_SHA`
	`SSL_DH_anon_EXPORT_WITH_RC4_40_MD5`
	`SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA`
	`SSL_RSA_WITH_NULL_SHA`
	`SSL_RSA_WITH_NULL_MD5`
	Default Value: None.
		`http.sender.cipher_suites=SSL_RSA_WITH_NULL_SHA,SSL_RSA_WITH_3DES_EDE_CBC_SHA`

http.sender.customizer_class	Specifies the class name for customizing the HTTP sender. Default Value: `oracle.oai.agent.adapter.technology.HTTPDefaultSenderCustomizer`	`http.sender.customizer_class=MyHTTPSenderCustomizer`
http.sender.password	Specifies the password used in the sender.password authentication. This password can also be encrypted by running the encrypt tool and renaming this parameter to encrypted_http.sender.password. Default Value: None. **See Also:** "How do I secure my passwords?" for instructions on how to modify and retrieve the password	`http.sender.password=httpuser`
http.sender.proxy_host	Specifies the proxy hostname. Default Value: None.	`http.sender.proxy_host=www-proxy.foo.com`
http.sender.proxy_port	Specifies the port number for the proxy host. This is needed if the proxy host is set. Default Value: None.	`http.sender.proxy_port=80`
http.sender.realm	Specifies the realm for the authentication scheme. Default Value: None.	`http.sender.realm=ipt`

Table 2–9 (Cont.) HTTP Adapter-specific Parameters

Parameter	Description	Example
http.sender.timeout	Specifies the time out for an HTTP connection. The unit is milliseconds. Default Value: `60000` milliseconds (60 seconds).	`http.sender.timeout=10000`
http.sender.username	Specifies the authentication user name. Default Value: None.	`http.sender.username=joe`
http.sender.wallet_ location	Specifies the path and name of the exported wallet file (not `.p12` file). This is required only when SSL is used. Default Value: None.	`http.sender.wallet_ location=/private/foo/ certdb.txt`
http.sender.wallet_ password	Specifies the password for Oracle Wallet Manager. This is required only when SSL is used. This password can also be encrypted by running the encrypt tool and renaming this parameter to encrypted_http.sender.wallet_ password. Default Value: None. **See Also:** "How do I secure my passwords?" for instructions on how to modify and retrieve the password	`http.sender.wallet_ password=walletuser`
ota.d3ls	Specifies the list of D3L XML files used by the bridge. Each business event handled by the bridge must have its own D3L XML file. When a new D3L XML file is imported in iStudio for use by an application using the HTTP adapter, the parameter must be updated and the HTTP adapter restarted.	`ota.d3ls=person.xml, person1.xml`
ota.receive. endpoint	Specifies the URL of the receiving application. The URL is of the form: `http[s]://hostname:port/path` Default Value: None.	`ota.receive.endpoint= http://site.com:8888/ servlet/inbound`
ota.send.endpoint	Specifies the URL of the sending application. The URL is of the form: `http[s]://hostname:port/path` Default Value: None.	`ota.send.endpoint=` `http://site.com:8888/servl et/inbound`
ota.type	Specifies the message payload type that the HTTP adapter handles for both incoming and outgoing messages. The options are `XML`, `XML_ NVP`, or `D3L`. Default Value: `XML`.	`ota.type=XML`

3

Design Time and Runtime Concepts

This chapter describes the design time and runtime concepts for the HTTP adapter. It contains the following topics:

3.1 HTTP Adapter Design Time Concepts

The HTTP adapter can handle XML and D3L structured payloads, such as pure XML data with strings beginning with <xml..., and binary data described by a D3L XML file.

3.1.1 XML Payload Type

You can import a Document Type Definition (DTD) in iStudio to determine how the HTTP adapter parses a received XML document into an OracleAS Integration InterConnect application view event. In addition, you can use the DTD to describe how an inbound application view message is converted to an XML document. Use the message type option XML when defining a new integration point in any of the event wizards.

Ensure that the `ota.type` parameter in the `adapter.ini` file is set to `XML` or `XML_NVP`, instead of `D3L`. Both the `XML` and `XML_NVP` settings operate with XML messages.

`XML` and `XML_NVP` differ in that `XML_NVP` supports legacy applications where the body of the HTTP message is prepended with the string, `message=`.

When the HTTP adapter operates in the XML payload mode, no transformations are performed on the messages between native view and application view. Any Extensible Stylesheet Language Transformations (XSLT) should be performed either before sending or receiving an XML document to or from OracleAS Integration InterConnect.

3.1.2 D3L Payload Type

The HTTP adapter supports both XML and D3L data types. The HTTP adapter performs a two-way conversion and transformation of messages between application view and native format.

An application based on the HTTP adapter can use the iStudio Message Type D3L and the iStudio D3L Data Type Import options when importing a data type. In this case, messages received or sent by the HTTP adapter must adhere to the fixed byte-level layout defined in a D3L XML file.

The D3L Data Type Import option can also define common view data types.

> **See Also:** Appendix B, *OracleAS Integration InterConnect User's Guide*

3.2 HTTP Adapter Runtime Concepts

This section describes the key runtime components of the HTTP adapter. This section contains the following topics:

- HTTP Receiver
- HTTP Sender
- HTTP Adapter Message Format
- HTTP Message Headers
- HTTP Receiver Diagnostics

3.2.1 HTTP Receiver

The HTTP adapter receives incoming messages from a single receiving endpoint, which is a servlet provided by the HTTP adapter, serving the `POST` requests from HTTP clients to OracleAS Integration InterConnect.

In a typical deployment scenario, the servlet runs in Oracle Application Server Containers for J2EE (OC4J). The servlet processes the HTTP client requests and sends them to the HTTP receiver through RMI. When the message is received, the HTTP receiver passes the message to the HTTP bridge.

The HTTP bridge uses the D3L XML file based on name/value pairs or magic value message header attributes (a sequence of bytes in the native format message header). The HTTP bridge uses this information to parse from the native message to an OracleAS Integration InterConnect message object and translate it to an application view event. The agent converts the application view event to a common view event and sends it to OracleAS Integration InterConnect for further routing and processing.

In the publish/subscribe mode, after the message is successfully sent to OracleAS Integration InterConnect, the HTTP adapter returns an acknowledgment message of type `200`.

In the synchronous request/reply mode, if the incoming message is a request, then the adapter will send back a reply instead of sending an acknowledgement message, whereas in the asynchronous request/reply mode, the adapter will send the acknowledgement message and the reply will go to the send endpoint defined by the `ota.send.endpoint` parameter.

The properties for the HTTP receiver are defined in the `adapter.ini` file and take the form, `http.receiver.*`.

See Also:

- *Oracle Application Server Integration InterConnect User's Guide*, Appendix B, for additional information on D3L name/value pair and magic value message header attributes
- Figure 1–1, "Incoming Messages" on page 1-2

3.2.2 HTTP Sender

The HTTP adapter consists of the HTTP bridge and runtime agent. When the agent has a message to send to an endpoint, the bridge is notified. The bridge then uses D3L XML to translate the common view object to the native format message. The native format message is then sent through the HTTP transport layer to an HTTP endpoint. In the request/reply mode, if the outgoing message from the hub is a request, then the bridge will wait for a reply from the remote HTTP server. After the reply is received, it will be passed on to the hub. If there is no reply within 60 seconds, which is the default time, then the request will be timed out. The default time can be modified using the `http.sender.timeout` parameter. The properties for the HTTP sender are defined in the `adapter.ini` file and take the form `http.sender.*`.

The HTTP adapter supports sending outgoing messages from OracleAS Integration InterConnect to multiple HTTP endpoints. The multiple endpoints feature enables sending messages to various remote Web servers.

An endpoint is associated with a subscribing event in iStudio by adding the transport properties such as the HTTP endpoint as metadata for the event. This is done using the Modify Fields function of the Subscribe Wizard - Define Application View dialog. After associating an endpoint and event, the message from the subscribing event is sent to the HTTP endpoint.

When using the multiple endpoint feature with XML data type, use the Generic event type, instead of XML. Using the Generic event type allows you to enter the metadata for the endpoints using the Modify Fields feature associated with iStudio.

Table 3–1 shows how metadata is associated with an event called `sendOrder` that sends an order to an HTTP server at `foo.com` with a path, `/servlet/test`.

Table 3–1 SendOrder Event Metadata

Parameter	Description
ota.endpoint=sendOrderAppEP	Specifies a unique endpoint name set in iStudio
ota.send.endpoint=http://foo.com/servlet/test	Specifies the sending endpoint for the HTTP adapter

If no metadata is associated with an event, then the endpoint specified by the `ota.send.endpoint` parameter in the `adapter.ini` file is used as the default endpoint.

Note: The `sender` properties are not inherited from the `adapter.ini` file.

See Also:

- Figure 1–2, "Outgoing Messages" on page 1-3
- Chapter 4 of the *Oracle Application Server Integration InterConnect User's Guide* for information on adding transport properties as metadata in iStudio

3.2.3 HTTP Adapter Message Format

This section describes how to extract or send messages from and to the HTTP adapter using different payload types. The HTTP adapter expects all payload types to be sent using the `POST` method, which does not have the `GET` method's data length limitations. This section includes the following topics:

- D3L Payload Type
- XML Payload Type
- XML_NVP (XML Name-Value Pair) Payload Type

3.2.3.1 D3L Payload Type

The ota.type parameter in the `adapter.ini` file must be set to `D3L` to use this payload type. The HTTP adapter receives a message from an HTTP client using the `POST` method. The data received with the `POST` method is interpreted as the payload. The HTTP adapter sends the payload using the `POST` method to one of the following endpoints:

- The endpoint associated with the event , if one is given
- The default endpoint specified by the `ota.send.endpoint` parameter in the `adapter.ini` file

3.2.3.2 XML Payload Type

The `ota.type` parameter in the `adapter.ini` file must be set to `XML` to use this payload type. The sending and receiving operation for the XML payload type is similar to that of D3L.

3.2.3.3 XML_NVP (XML Name-Value Pair) Payload Type

The `ota.type` parameter in the `adapter.ini` file must be set to `XML_NVP` to use this payload type. The HTTP adapter expects the payload to be packaged in the following manner:

```
application= ..&...&message=<?xml .
.
.
>
```

The value of the message name/value pair contains the payload. During the receiving operation, the HTTP adapter extracts the message name/value pair from the `POST` data and converts it to an OracleAS Integration InterConnect object. During the send operation, the adapter packages the name/value pair and sends it through the `POST` method.

See Also: The `ota.type` parameter description in Table 2–9 for information on setting the payload message type in the `adapter.ini` file

3.2.4 HTTP Message Headers

The HTTP adapter uses custom message headers, in addition to the default message headers. Example 3–1 shows the HTTP message header types and the data sent by the HTTP adapter:

Example 3–1 HTTP Message Header Types and Data

```
OAI-MV = QA/V1 (Message Version)
CONNECTION = Keep-Alive, TE
CONTENT-TYPE = application/octet-stream
USER-AGENT = RPT-HTTPClient/0.3-2S
OAI-T = 0
OAI-BO = Persona
OAI-EV = QA/V1
TE = trailers, deflate, gzip, compress
ACCEPT-ENCODING = deflate, gzip, x-gzip, compress, x-compress
OAI-EN = newPerson1a (Event name)
CONTENT-LENGTH = 76
HOST = cc-sun.us.oracle.com:8888
OAI-APPLICATION = HTTP1A
```

The `OAI-*` headers are associated with a specific HTTP adapter. This information is useful in debugging and tracking. Table 3–2 describes the key `OAI-*` headers.

Table 3–2 OAI-* Headers

Header	Description
`OAI-APPLICATION`	HTTP adapter application name
`OAI-BO`	Business object name to which this message corresponds
`OAI-EN`	Oracle Application Server Integration InterConnect event name
`OAI-EV`	Event version to which this message corresponds, as created in iStudio
`OAI-MV`	Message version to which this message corresponds, as created in iStudio
`OAI-T`	Tag that represents the mode of the adapter that publishes, requests or responds to a message Possible values are: `0` for publish `1` for request `2` for reply

3.2.5 HTTP Receiver Diagnostics

To determine whether the HTTP receiver is functioning properly, perform the following steps:

1. Open a Web browser.
2. Enter the URL specified for the `ota.receive.endpoint` parameter in the `adapter.ini` file.

   If the servlet is deployed properly, then the Web browser displays information similar to the following:

Please use HTTP POST to send request.

HOST	cchung-sun:8889
CONNECTION	keep-alive
USER-AGENT	Mozilla/4.7 [en] (WinNT; I)
ACCEPT	image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
ACCEPT-ENCODING	gzip
ACCEPT-LANGUAGE	en
ACCEPT-CHARSET	iso-8859-1,*,utf-8
VIA	1.0 inet-netcache2 (NetCache NetApp/5.1R2D10)
X-FORWARDED-FOR	144.25.140.180

rmiHost	localhost
rmiPort	9901
instanceName	ip
logging	on
log level	debug
log file	/private1/cchung/oc4j/j2ee/home/ts_ip_1010801927749.log

certs=

This page helps identify information that the servlet reads from the `web.xml` file. The rmiHost, rmiPort, and instance name must match the corresponding parameters in the `adapter.ini` file.

3.3 Customizing the HTTP Adapter

You can customize the adapter behavior by implementing the following interfaces:

- `oracle.oai.adapter.agent.technology.ReceiverCustomizer`
- `oracle.oai.adapter.agent.technology.HTTPSenderCustomizer`

3.3.1 ReceiverCustomizer Interface

You can use the `ReceiverCustomizer` interface to customize the `TransportMessage` object that HTTP adapter receives. The `TransportMessage` object represents the native message that the transport layer receives or sends.

- If you wish to customize the `TransportMessage` object itself, use the `customizeTransportMessage()` method. This method is called before the before the adapter processes the `TransportMessage` object.
- If you wish to modify the message itself, then implement the `customizeTransportMessage()` method. You must also implement the `createReplyMessage()` method and ensure that it returns a `null` value.

The following code describes the file structure of the `ReceiverCustomizer` interface.

```
package oracle.oai.agent.adapter.technology;
import oracle.oai.agent.adapter.transport.TransportMessage;
import oracle.oai.agent.adapter.sdk.Agent;
public interface ReceiverCustomizer {
public void customizeTransportMessage(Agent agent, int receiverType,
TransportMessage transportMessage);
```

```
public String createReplyMessage(Agent agent,int status, TransportMessage
receivedTransportMessage, String response)
}
```

The following table summarizes the `ReceiverCustomizer` interface.

Method	Description
`customizeTransport Message();`	Allows you to customize the transport message received by the adapter. It uses the following parameters: agent: Log a message. receiverType: Information on the type of adapter. transportMessage: Customize the transport message received by the adapter.
`createReplyMessage ();`	This method creates a reply message based on the status and message received. It contains the following parameters: `agent`: Log a message. `status`: Status of the message process. If the value is TransportResponse.TRANSPORT_ACK, then the message is processed successfully. If the value is TransportResponse.TRANSPORT_ERROR, then the message is not processed successfully. `receivedTransportMessage`: Transport message received by the adapter. This parameter is used to transport headers in the transport message to create a meaningful HTTP message. `response`: Response returned by the agent for a request sent by the client, in the request/reply scenario. You can customize this response to create the reply message that will be sent back to the client. The return string contains the reply message. This method is included for backward compatibility.

Example 3–2 customizes the `MyReceiverCustomizer` class to remove the first line in the native message.

Example 3–2 Example of ReceiverCustomizer

```
import oracle.oai.agent.adapter.sdk.Agent;
import oracle.oai.agent.adapter.transport.TransportMessage;
import oracle.oai.agent.adapter.transport.TransportException;
import oracle.oai.agent.adapter.technology.ReceiverCustomizer;

public class MyReceiverCustomizer implements ReceiverCustomizer {

// This example describes how to remove an extra line from a file that
// OracleAS Integration InterConnect does not understand.

public void customizeTransportMessage(Agent agent, int receiverType,
                                      TransportMessage transportMessage) {
   String payload = transportMessage.getBodyAsString();

//For debugging purposes only, the following syntax removes the first line from
the payload. Details of removeFirstLine() is not provided.
      agent.logTraceMessage("payload received = " + payload, null, null, null);
      String newPayload = removeFirstLine(payload);
         try {
            transportMessage.setBody(newPayload);
         }
         catch(TransportException te) {
         . . . .
```

```
      }
   }

   public String createReplyMessage(Agent agent, int status, TransportMessage
receivedTransportMessage, String response) {
      String response = Message has unknown status.";
      switch (status) {

//OracleAS Integration InterConnect indicates to the transport layer that the
// message has been processed successfully.

      case TransportResponse.TRANSPORT_ACK:
         return "Request has been processed successfully.";

//OracleAS Integration InterConnect indicates to the transport layer that the
// message cannot be processed successfully.

      case TransportResponse.TRANSPORT_ERROR:
         return "Please try again. The server cannot process your request.";
   }
   return "Message has unknown status.";
}
```

List of Methods for the TransportMessage Class

The following table provides a list of methods you may choose for the `TransportMessage` class.

Method	Description
`public byte[] getBodyAsBytes();`	Get the body of the message as a byte array. Return the message in the byte array.
`public InputStream getBodyAsInputStream();`	Get the body of the message and return an `InputStream` object representing the body of the message.
`public Properties getTransportHeaders();`	Get all transport specific header and return a Properties object that contains all the transport headers.
`public String getBodyAsString();`	Get the body of the message as a String object. Return the message in the String object.
`public String toString();`	Dump message and headers.
`public void setBody(InputStream in) throws TransportException;`	Set the body of the message. The body type will be set to `BYTES`. The parameter is: `in`: Contains the message. It is an Inputstream object. It throws a `TransportException`.
`public void setBody(String body) throws TransportException;`	Set the body of the message. The body type will be set to `STRING`. The parameter is: `body`: body of the message It throws a `TransportException`.
`public void setTransportHeader(String name, String value);`	Set a transport specific header.

3.3.2 HTTPSenderCustomizer Interface

You can use the `HTTPSenderCustomizer` interface to customize the subject name and payload of the `TransportMessage` object that is sent to the transport layer. The `HTTPSenderCustomizer` interface extends the `SenderCustomizer` interface.

3.3.2.1 SenderCustomizer Interface

The following code describes the file structure of the `SenderCustomizer` interface.

```
package oracle.oai.agent.adapter.technology;
import oracle.oai.agent.adapter.sdk.MessageObject;
import oracle.oai.agent.adapter.sdk.AttributeObject;
import java.util.Properties;
import oracle.oai.agent.adapter.sdk.Agent;
import oracle.oai.agent.adapter.transport.TransportMessage;

public interface SenderCustomizer {

public void customizeTransportMessage(Agent agent,
                                       TransportMessage transportMessage,
                                       MessageObject mobj,
                                       AttributeObject aobj);

}
```

customizerTransportMessage method

This method specifies how to customize the transport message for the transport sender. The adapter creates a TransportMessage object for the transport layer to send, based on the MessageObject object sent by OracleAS Integration InterConnect. You can use this method to further customize the transport message to be sent by the transport layer.

This method contains the following parameters:

agent: Log messages.

transportMessage: The `TransportMessage` object that the adapter has created for sending.

mobj: The `MessageObject` object parsed from OracleAS Integration InterConnect.

aobj: The `AttributeObject` object parsed from OracleAS Integration InterConnect.

This method does not return anything. You can change the payload with the `TransportMessage` parameter.

3.3.2.2 HTTPSenderCustomizer Interface

The following code describes the file structure of the `HTTPSenderCustomizer` interface.

```
package oracle.oai.agent.adapter.technology;
import oracle.oai.agent.adapter.sdk.MessageObject;
import oracle.oai.agent.adapter.sdk.AttributeObject;

   public interface HTTPSenderCustomizer extends SenderCustomizer{

}
```

3.4 Starting the HTTP Adapter

The process for starting the adapter varies based on the operating system.

- To start the HTTP adapter on UNIX:
  1. Change to the directory containing the start script.

     ```
     cd ORACLE_HOME/integration/interconnect/adapters/Application
     ```

  2. Type **`start`** and press **Enter**.
- To start the HTTP adapter from Services on Windows:
  1. Access the Services window from the Start menu. The Services window is displayed.

On...	Choose...
Windows 2000	Start, Settings, Control Panel, Administrative Tools, Services

  2. Select the **OracleHomeOracleASInterConnectAdapter-Application** service.
  3. Start the service based on the operating system.

On...	Choose...
Windows 2000	Right-click the service and choose Start from the context menu.

> **Note:** You can also start and stop the HTTP adapter using the IC Manager. Refer to *Oracle Application Server Integration InterConnect User's Guide* for more details.

3.4.1 Log File of HTTP Adapter

You can verify the startup status of the HTTP adapter by viewing the `oailog.txt` files. The files are located in the timestamped subdirectory of the `log` directory of the HTTP adapter. Subdirectory names have the following form:

```
timestamp_in_milliseconds
```

The following is an example of the information about an HTTP adapter that started successfully:

```
The Adapter service is starting..
Registering your application (HTTPAPP)..
Initializing the Bridge oracle.oai.agent.adapter.technology.TechBridge..
Starting the Bridge oracle.oai.agent.adapter.technology.TechBridge..
Service started successfully.
```

3.5 Stopping the HTTP Adapter

Based on the operating system, the process for stopping the adapter varies.

- To stop the HTTP adapter on UNIX:
  1. Change to the directory containing the stop script.

     ```
     cd ORACLE_HOME/integration/interconnect/adapters/Application
     ```

2. Type **`stop`** and press **Enter**.

- To stop the HTTP adapter from Services on Windows:

1. Access the Services window from the Start menu. The Services window is displayed.

On...	Choose...
Windows 2000	Start, Settings, Control Panel, Administrative Tools, Services

2. Select the **OracleHomeOracleASInterConnectAdapter-Application** service.
3. Stop the service based on the operating system.

On...	Choose...
Windows 2000	Right-click the service and choose Stop from the context menu.

You can verify the stop status of the HTTP adapter by viewing the `oailog.txt` files. These files are located in the timestamped subdirectory of the `log` directory of the HTTP adapter.

A

Frequently Asked Questions

This chapter provides answers to frequently asked questions about the HTTP adapter.

How do I know whether the HTTP adapter has started properly?

View the `oailog.txt` file located in the timestamped subdirectory of the HTTP adapter `logs` directory.

On...	Change to...
UNIX	*ORACLE_HOME*`/integration/interconnect/adapters/`*Application*`/logs/`*timestamp_in_milliseconds*
Windows	*ORACLE_HOME*`\integration\interconnect\adapters\`*Application*`\logs\`*timestamp_in_milliseconds*

where *Application* is the value you defined in Step 4 on page 2-2, and *timestamp_in_milliseconds* is the directory. If no exceptions are listed, then the adapter has started properly.

The HTTP adapter did not start properly. What is wrong?

View the exceptions in the adapter log file (`oailog.txt`). The exceptions provide information about inconsistencies. One possible reason is that the HTTP adapter did not connect to the repository. Ensure that the repository is started properly and the HTTP adapter connects to the repository once it is started properly. You do not need to restart the adapter.

See Also: *Oracle Application Server Integration InterConnect User's Guide* for instructions on starting the repository on UNIX and Windows

I changed an element in iStudio, but the HTTP adapter uses old information. What is happening?

The HTTP adapter caches information from iStudio. The information is stored locally in the repository. If you change something in iStudio and want to view the change in the runtime environment, then you need to perform the following procedure:

1. Stop the adapter.
2. Delete the adapter cache files.
3. Restart the adapter.

Each adapter has a `persistence` directory located in the directory named after the HTTP application. Deleting this directory when the adapter has been stopped makes the adapter obtain the new metadata from the repository when started.

If I cannot answer some HTTP configuration questions or I make a mistake during installation, can I edit these settings later?

Yes, you can edit the parameters in the following file:

On...	Change to...
UNIX	*ORACLE_HOME*`/integration/interconnect/adapters/`*Application*`/adapter.ini`
Windows	*ORACLE_HOME*`\integration\interconnect\adapters\`*Application*`\adapter.ini`

Note: All configuration parameters with the exception of `bridge_class` can be edited more than once.

See Also: "hub.ini Files" on page 2-8 for parameter information

Can I install multiple HTTP adapters on the same computer?

You can install multiple HTTP adapters on the same computer by specifying a different Oracle home for each adapter during the installation process. If you try to install a second adapter in the same Oracle home, then the installer overwrites previous installations of the HTTP adapter. However, you can still install multiple adapters in the same Oracle home, by using the `copyAdapter` utility and manually editing some configuration files.

How do I install a second HTTP adapter in the same Oracle home?

To install a second HTTP adapter in the same Oracle home, perform the following steps:

1. Use the `copyAdapter` utility to make a copy of the existing HTTP adapter:

   On UNIX:

   ```
   % cd ORACLE_HOME/integration/interconnect/bin
   ```

```
% copyAdapter oldAdapterName newAdapterName
```

On Windows:

```
c:\> cd ORACLE_HOME\integration\interconnect\bin
c:\> copyAdapter oldAdapterName newAdapterName
```

2. Change the parameters in the `adapter.ini` file for the new adapter. Ensure the parameters in the new `adapter.ini` file are different from the `adapter.ini` file for the existing HTTP adapter, as follows:
   a. Change the send endpoint (`ota.send.endpoint`) parameter.
   b. Change the receive endpoint (`ota.receive.endpoint`) parameter.

      The default receive endpoint set by the installer is:

      `http://machine name:port number/oai/servlet/transportServlet`

      You can change the receive endpoint to the following:

      `http://machine name:portnumber/oai1/servlet/transportServlet`

   c. Change the payload type parameter (`ota.type`), if necessary.
   d. Change the RMI registry port parameter (`http.receiver.registry_port`) to a port not used on this computer.
3. Change the content of the `web.xml` file to match that of the `adapter.ini` file. The `web.xml` file is in the following directory:

   On UNIX:

   `ORACLE_HOME/integration/interconnect/adapters/newAdapterName/webapps/WEB-INF`

   On Windows:

   `ORACLE_HOME\integration\interconnect\adapters\newAdapterName\webapps\WEB-INF`

4. Change the RMI port to match the value entered in Step 2d.
5. Change the following entry in the `web.xml` file from:

   ```
   <param-value>9901</param-value>
   ```

   to:

   ```
   <param-value> port-number-you-used-step-2d </param-value>
   ```

6. Change the following entry in the `application.xml` file in the `ORACLE_HOME\integration\interconnect\adapters\<your new http app name>\webapps\META-INF` directory:

   ```
   <context-root>oai/servlet</context-root>
   ```

   to:

   ```
   <context-root>oai1/servlet</context-root>
   ```

7. Create the Java archive parameter (`oai1.ear`):

   On UNIX:

   ```
   % cd ORACLE_HOME/integration/interconnect/adapters/<your http app name>/webapps
   % jar cvf oai.war WEB-INF
   % jar cvf oai1.ear oai.war META-INF
   ```

On Windows:

```
c:\> ORACLE_HOME\integration\interconnect\adapters\<your new http app
name>\webapps
c:\> jar cvf oai.war WEB-INF
c:\> jar cvf oai1.ear oai.war META-INF
```

An `.ear` file called `oai1.ear` has been created, which is ready for deployment.

8. Deploy the `oai1.ear` file in the OracleAS environment:

   On UNIX:

```
% cd ORACLE_HOME/dcm/bin
% dcmctl shell
dcmctl> deployApplication -f oai1.ear  -a oaiservlet1 -co oc4j_oai
dcmctl> exit
```

   On Windows:

```
c:\> ORACLE_HOME\dcm\bin\dcmctl shell
dcmctl> deployApplication -f oai1.ear  -a oaiservlet1 -co oc4J_OAI
dcmctl> exit
```

> **Note:** `oaiservlet1` is a unique application name that you assign to your servlet. If this name is already used in the current environment, then select a different name.

Restart the HTTP server. Verify whether the new receiving endpoint is functioning by entering the URL used in Step 2b. If the servlet is deployed correctly, then a diagnostic page is displayed.

How do I secure my passwords?

OracleAS Integration InterConnect uses Oracle Wallet Manager to maintain system passwords. When you install OracleAS Integration InterConnect, Oracle Wallet Manager is also installed and a password store is created. All passwords used by OracleAS Integration InterConnect components are stored in the password store. The password is stored in the Oracle Wallet in the following format:

```
ApplicationName/password
```

For example,

```
AQAPP/aq_bridge_schema_password
```

The `ApplicationName` is the name of the application, which is extracted from the `adapter.ini` file of the corresponding adapter. In the `adapter.ini` file, the `application` parameter specifies the `ApplicationName` to which this adapter connects. The password for the application is also retrieved from the `adapter.ini` file.

You can create, update, and delete passwords using the `oraclewallet` command. When you run the command, it prompts you for the admin password.

You can use the following commands to manage your passwords:

- List all passwords in the store

```
oraclewallet -listsecrets
```

- Create a password

  ```
  oraclewallet -createsecret passwordname
  ```

  For example, to create a password for the hub schema:

  ```
  oraclewallet -createsecret hub_password
  ```

- View a password

  ```
  oraclewallet -viewsecret passwordname
  ```

  For example, to view the password for the hub schema:

  ```
  oraclewallet -viewsecret hub_password
  ```

- Update a password

  ```
  oraclewallet -updatesecret passwordname
  ```

  For example, to update the password for the hub schema:

  ```
  oraclewallet -updatesecret hub_password
  ```

- Delete a password

  ```
  oraclewallet -deletesecret passwordname
  ```

  For example, to delete the password for the hub schema:

  ```
  oraclewallet -deletesecret hub_password
  ```

B

Example of the adapter.ini File

This appendix shows an `adapter.ini` example file for the HTTP adapter.

> **See Also:** "Configuring the HTTP Adapter" on page 2-7 for additional information on `adapter.ini` configuration parameters

This section shows an `adapter.ini` example file for the HTTP adapter.

```
#include <../../hub/hub.ini>
// *************
// ** Adapter **
// *************
// Application (as created in iStudio) corresponding to this Adapter.
application=HTTPapp1
// Partition (as created in iStudio) corresponding to this Adapter.
partition=
// If you want to have multiple adapter instances for a given application with the
// given partition, each Adapter should have an instance number.
// instance_number=2
// Bridge class
bridge_class=oracle.oai.agent.adapter.technology.TechBridge

//-----------------------------------------
// HTTP Adapter Endpoint information
//-----------------------------------------
// time out in milli seconds (default should be set to 60000 milli seconds)
// This is used to time-out a http connection. Use default.
// http.sender.timeout=

// set the following if authentication is needed.
// authentication type (Valid options: basic or digest)
http.sender.authtype= basic
http.sender.realm=ipt
http.sender.username=scott
encrypt_http.sender.password=1124110710711065108010941084107310701071108110 69

// set the proxy parameters if proxy is needed.
http.sender.proxy_host=www-proxy.test.com
http.sender.proxy_port=80


// set the security parameters if SSL is used.
http.sender.wallet_location=certdb.txt
encrpyt_http.sender.wallet_
password=1124110710711065108010941084107310701071108110 70
//
```

```
//   If this is not set, we will use the
//   default ciphers suites provided by
//   SSLSocketFactory.
//   The selections are:
//                      SSL_RSA_WITH_3DES_EDE_CBC_SHA
//                      SSL_RSA_WITH_RC4_128_SHA
//                      SSL_RSA_WITH_RC4_128_MD5
//                      SSL_DH_anon_WITH_3DES_EDE_CBC_SHA
//                      SSL_DH_anon_WITH_RC4_128_MD5
//                      SSL_DH_anon_WITH_DES_CBC_SHA
//                      SSL_RSA_WITH_DES_CBC_SHA
//                      SSL_RSA_EXPORT_WITH_RC4_40_MD5
//                      SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
//                      SSL_DH_anon_EXPORT_WITH_RC4_40_MD5
//                      SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA
//                      SSL_RSA_WITH_NULL_SHA
//                      SSL_RSA_WITH_NULL_MD5
// Use "," as delimiter. An example cipher suites  is:
//  SSL_RSA_WITH_3DES_EDE_CBC_SHA,SSL_RSA_WITH_NULL_SHA
//
//http.sender.cipher_suites=



//----------------------------------------
// HTTP Receiver initialization variables
//----------------------------------------

// rmi port used by http receiver (default is 1099)
http.receiver.registry_port = 1099

// instance name to distinguish other instances of receiver
http.receiver.instance_name =oai


// A list of the D3L XML files used by this Bridge. Each business event handled
// by the Bridge must have it's own D3L XML file.
// Whenever a new D3L XML file has been imported in iStudio to be used by
// an application using the HTTP adapter, the following parameter must
// be updated and the adapter restarted.
ota.d3ls=person.xml, person1.xml


// *************
// ** Agent  ***
// *************

// Log level (0 = errors only, 1 = status and errors, 2 = trace, status and
errors).
agent_log_level=2

// Hub message selection information
agent_subscriber_name=HTTPapp1
agent_message_selector=recipient_list like '%,HTTPapp1,%'
// Only provide values for the next two parameters if you have multiple Adapter
// instances for the given application with the given partition.
//agent_reply_subscriber_name=
//agent_reply_message_selector=

// Set this to false if you want to turn off all tracking of messages (if true,
```

```
// messages which have tracking fields set in iStudio will be tracked)
agent_tracking_enabled=true

// Set this to false if you want to turn off all throughput measurements
agent_throughput_measurement_enabled=true

// By default, adapters use an OAI specific DTD for all messages sent to the Hub
// because other OAI adapters will be picking up the messages from the Hub
// and know how to interpret them. This should be set to true if for every
// message,you would like to use the DTD imported for that message's Common View
// instead of the OAI DTD.  This should only be set to true if an OAI Adapter is
// *NOT* receiving the messages from the Hub.
agent_use_custom_hub_dtd=false

// Sets the metadata caching algorithm. The possible choices are startup (cache
// everything at startup: this may take a while if there is a lot of metadata
// in your Repository), demand (cache metadata as it is used) or none (no caching
//: this will slow down performance.)
agent_metadata_caching=demand

// Sets the DVM table caching algorithm.  The possible choices are startup (cache
// all DVM tables at startup: this may take a while if there are a lot of tables
// in your Repository), demand (cache tables as they are used) or none (no caching
//: this will slow down performance.)
agent_dvm_table_caching=demand

// Sets the lookup table caching algorithm.  The possible choices are startup
// (cache all lookup tables at startup: this may take a while if there are
// a lot of tables in your Repository), demand (cache tables as they are used) or
// none (no caching: this will slow down performance.)
agent_lookup_table_caching=demand

// If metadata caching, DVM table caching, or lookup table caching are turned on
// (startup or demand) then the Adapter caches metadata or DVM tables it retrieves
// from the Repository in a file cache.  When you restart the Adapter, it will not
// have to get that metadata or DVM table from the Repository again because it is
// in the cache files.  However, if you change some metadata or DVM table
// using iStudio and you want the Adapter to use those changes the next time it is
// started, you can either delete the cache files or set this parameter to true
// before restarting.
agent_delete_file_cache_at_startup=false

// Max number of application data type information to cache
agent_max_ao_cache_size=200

// Max number of common data type information to cache
agent_max_co_cache_size=100

// Max number of message metadata to cache
agent_max_message_metadata_cache_size=200

// Max number of DVM tables to cache
agent_max_dvm_table_cache_size=200

// Max number of lookup tables to cache
agent_max_lookup_table_cache_size=200

// Internal Agent queue sizes
agent_max_queue_size=1000
agent_Persistence_queue_size=1000
```

```
// Persistence
agent_persistence_cleanup_interval=60000
agent_persistence_retry_interval=60000
```

Index

I

L

M

O

P

R

S

T

W

X